Proc. of the European Safety and Reliability Conference (ESREL 2026)
Edited by José C. Matos, Paulo B. Lourenço, Michael Beer, Daniel Oliveira,
Edoardo Patelli, Hélder S. Sousa, Rui A. Silva and Monica Santamaria
 Published by Research Publishing, Singapore.
doi: 10.3850/ESREL2026061419_esrel26-p26348-cd

What We Know We Don't Know: Systematic Risk Identification from Contradictory Safety Documentation

Masoud Ebrahimi[1,2], Kaj Hänninen[2], Kristina Lundqvist[2], and Marjan Sirjani[2]
[1] *Formal Trust AI, Sweden. E-mail: masoud.ebrahimi@formaltrust.ai*
[2] *Department of Computer Science & Engineering, Mälardalen University, Sweden.*

Ensuring accountability and responsibility in human-centered safety-critical systems, such as autonomous vehicles and medical devices, is essential. These systems must assure safety, comply with normative constraints, ethical principles, and regulatory standards while navigating complex human-machine interactions. Existing methods fall short in verifying normative properties and attributing responsibility for many reasons. Insufficient knowledge of the unknown risks that remain in systems being one of them. We propose a Kripke-based approach that formally connects the Open World Assumption of ontological Knowledge Graph with Closed World Assumption analysis for systematic risk identification. The approach presented can increase knowledge of the unidentified risks in the set of the closed world assumptions of the total residual risk. Our methodology addresses the challenge that Knowledge Graph extraction from informal documentation is not formulated precisely enough to be validated or verified, preventing automation and correctness guarantees. We construct Kripke structures where each state represents a world model satisfying Knowledge Graph configurations derived from contradictory documentation. The framework can integrate expert feedback, ensuring alignment with domain knowledge and regulatory expectations. A case study on a simplified automotive seatbelt warning system demonstrates the framework's ability to compare documented safety properties against latent hazard criteria and reveals a 40.9% coverage gap: states classified as safe by specification but still harboring residual risks due to sensor-actuator conflation and unobservable safety properties.

Keywords: Unknown risks identification, Open-World Assumption, Known Unknowns, safety-critical systems, accountability and responsibility.

1. Introduction

Safety-critical systems are governed by documentation of requirements, design standards, operational procedures, and regulatory compliance artifacts (Alur, 2015; ISO, 2018). This documentation is incomplete, uses modal language, and contains contradictory statements when viewed from different operational contexts (Palshikar, 2001). Documentation management for safety analysis is a challenge (Hall et al., 2017). Accountability attribution further complicates this process.

Knowledge Graphs (KGs) structure and integrate documents of heterogeneous sources by representing entities as nodes and relationships as edges (Ehrlinger and Wöß, 2016). They enable semantic queries, automated reasoning, and traceability while operating under the *Open World Assumption (OWA)*, where the absence of a fact does not imply its negation (Gruber, 1993). However, KGs do not capture modality: they encode static relationships but not temporal dynamics, state transitions, or modal distinctions for safety analysis. Constructing KGs from safety documentation faces three challenges: (i) modal statements are misinterpreted as conflated facts, leading to contradictions; (ii) incomplete documentation leaves relationships unspecified; and (iii) the lack of formal semantics prevents axiom violation identification and mitigation.

Although these issues are real, documentation-to-KG construction is empirically feasible through established ontology engineering and knowledge acquisition workflows that combine automatic extraction with expert curation (Musen, 2015; Ji et al., 2022). In safety and security engineering, related evidence structuring, traceability, and risk documentation practices are also well established (Leveson, 2012; Hall et al., 2017; Alur, 2015); our contribution starts from such extracted triples and adds formal modal validation to expose residual risks.

In contrast, certification mandates that systems meet safety properties under all conditions. Therefore, safety analysis requires the *Closed World Assumption (CWA)*, where every relevant fact must be assigned a truth value to enable formal verification (Clarke et al., 2018). The mismatch between OWA and CWA creates a *verification gap* in which residual risks remain hidden. When hazards arise from undocumented facts (*"known unknowns"*), responsibility attribution becomes unclear, and some safety gaps may be overlooked.

We propose a methodology to bridge OWA and CWA through *construction-time reasoning* that formally validates and corrects KGs. Contradictions are treated not as errors to be resolved, but as distinct world models to be enumerated. We interpret KGs as conjunctions over sets of triples. Starting from an initial world containing only entity types, we add relationships derived from documentation, searching for maximally determined states. If adding another relationship to a state leads to contradiction, it is a *maximally determined state*. Such states receive self-loops, representing terminal states.

This construction partitions contradictory statements by creating distinct worlds. Each world represents a coherent interpretation of the documentation. For a world with an undecided relationship (a *known unknown*), we branch it into two worlds: one where the relationship holds and one where it does not, exposing residual risks that are left unspecified. The resulting Kripke structure is complete when all reachable paths terminate in maximally determined states with self-loops.

We annotate stable terminals by safety properties, classifying them as safe or unsafe. The distance between unsafe and safe states quantifies the severity of residual risks: a small distance indicates isolated gaps in the specification, while large distances suggest systematic underspecification.

To keep the presentation readable, we first define the formal objects, then interpret their practical meaning in the case study and discussion.

Section 2 covers preliminaries. Section 3 proposes our methodology, the Kripke semantics for KGs, and their validation. Section 4 presents a case study. Section 5 compares with related work, and Section 6 concludes and outlines future work.

2. Preliminaries

Ontologies provide vocabulary for describing a domain. They define concepts (classes), relationships (properties), and constraints (axioms) to structure the domain knowledge.

Definition 2.1. An ontology $\mathcal{O} = \langle \mathcal{C}, \mathcal{R}, \mathcal{A} \rangle$ is a triplet of a finite set of concepts $\mathcal{C}$, a finite set of relations $\mathcal{R}$, and a finite set of axioms $\mathcal{A}$ defining constraints over concepts and relations.

Definition 2.2. Given ontology $\mathcal{O} = \langle \mathcal{C}, \mathcal{R}, \mathcal{A} \rangle$, we define $K = \langle N, E, \ell \rangle$ as a KG where N is a finite set of nodes representing entities, $E \subseteq N \times N$ is a set of directed edges representing relationships between entities (i.e., nodes), and ℓ is a function labeling entities (i.e., nodes) with concepts ($\ell : N \to 2^{\mathcal{C}}$) and edges with relations ($\ell : E \to 2^{\mathcal{R}}$). All instances and relationships in KG K must satisfy axioms in $\mathcal{A}$.

Definition 2.3. A Kripke structure over atomic propositions $\mathbb{AP}$ is defined as $M = \langle Q, I, R, L \rangle$ where Q is a finite set of states (worlds), $I \subseteq Q$ is a non-empty set of initial states, $R \subseteq Q \times Q$ is a transition relation, and $L : Q \to 2^{\mathbb{L}}$, where $\mathbb{L} = \mathbb{AP} \cup \{\neg p \mid p \in \mathbb{AP}\}$, is a labeling function assigning to each state a set of literals.

World. In this context, a *world* refers to a state $q \in Q$ in the Kripke structure M. We use world and state interchangeably to denote a specific truth valuations over atomic propositions $\mathbb{AP}$.

Reachability. For states $p, q \in Q$, we write $p \xrightarrow{*} q$ to denote that q is reachable from p. Formally, there exists a finite path $p = q_0, q_1, \ldots, q_n = q$ such that $\langle q_i, q_{i+1} \rangle \in R$ for all $0 \leq i < n$.

RDF Triples. We represent KGs using the RDF (Resource Description Framework) standard, encoding knowledge as *triples* $\langle s, p, o \rangle$ (subject, predicate, object). Each node's type is represented using a type-triple $\langle n, \mathrm{type}, c \rangle$ for node $n \in N$ and concept $c \in \ell(n)$, while an edge becomes a triple $\langle s, p, o \rangle$ for subject node $s \in N$, predicate $p \in \ell(\langle s, o \rangle)$, and object node $o \in N$. Finally, $\mathbb{S}$, $\mathbb{P}$, and $\mathbb{O}$ denote sets of all possible subjects, predicates, and objects.

The *alphabet* determines entities, and relations that can appear in the model. *Closed alphabet as-*

sumption, fixes finite sets of subjects $\mathbb{S}$, predicates $\mathbb{P}$, and objects $\mathbb{O}$ representing all possible terms. These sets capture the complete vocabulary available for expressing knowledge within the Kripke structure. The alphabet assumption is orthogonal to the CWA/OWA distinction: alphabet closedness concerns the vocabulary, while CWA/OWA concerns truth valuations over that vocabulary. In this work, we adopt a *closed alphabet assumption*.

3. Knowledge Graph Semantics

Safety documentation frequently contains modal statements (e.g., *"must activate"*, *"can occur"*), which express constraints, obligations, and possibilities rather than facts. Our semantics handles these modalities by partitioning them across distinct worlds, each representing a valid interpretation of the documentation. We define a formal semantics for KGs through Kripke structures that provides coverage guarantees by enumerating all possible truth valuations of undecided predicates.

Atomic Propositions and Literals. We convert each RDF triple $\langle s, p, o\rangle$ into predicate notation $p(s, o)$, treating it as an atomic proposition.

Definition 3.1. Given a KG $K = \langle N, E, \ell\rangle$ of an ontology $\mathcal{O} = \langle \mathcal{C}, \mathcal{R}, \mathcal{A}\rangle$, we construct the set of atomic propositions from two sources: node predicates P_N (representing entity types) and edge predicates P_E (representing entity relations). These are formally defined as:

- $P_N = \{\text{type}(n, c) \mid c \in \ell(n), n \in N\}$
- $P_E = \{p(s, o) \mid p \in \ell(\langle s, o\rangle), \langle s, o\rangle \in E\}$
- $\mathbb{AP} = P_N \cup P_E$

A literal is either an atomic proposition ap or its negation $\neg ap$. Universal literals set comprises atomic propositions and their negations; i.e., $\mathbb{L} = \mathbb{AP} \cup \{\neg p \mid p \in \mathbb{AP}\}$. Finally, $ap \in \mathbb{AP}$ is true in state q if $ap \in L(q)$, false if $\neg ap \in L(q)$, and undecided if neither ap nor $\neg ap$ appears in $L(q)$.

Formula notation. A KG K has a formula $\varphi(K)$ over atomic propositions, and worlds in Kripke M satisfying this formula are models of K.

3.1. *Kripke Construction*

Algorithm 1 enumerates all truth valuations over $\mathbb{AP}$ by exploring both valuations of edge predicates $p(s, o) \in P_E$, guaranteeing *coverage*: no feasible valuation is overlooked.

Algorithm 1 Kripke Structure Construction

```
1:  Input: P_N, P_E
2:  Output: M = ⟨Q, I, R, L⟩
3:  q_0 = ⋀ type(·) for type(·) ∈ P_N
4:  Q ← {q_0}, I ← {q_0}, R ← ∅
5:  queue ← {q_0}                          ▷ queue of states to explore
6:  while queue ≠ ∅ do
7:      q ← pop from queue
8:      extended ← false                   ▷ is extended flag
9:      for p(·) ∈ P_E where q ⊭ p(·) ∧ q ⊭ ¬p(·) do
10:         q⁺ ← q ∧ p(·), q⁻ ← q ∧ ¬p(·)
11:         if q⁺ ∉ Q then
12:             Q ← Q ∪ {q⁺}, R ← R ∪ {⟨q, q⁺⟩}
13:             queue ← queue ∪ {q⁺}, extended ← true
14:         if q⁻ ∉ Q then
15:             Q ← Q ∪ {q⁻}, R ← R ∪ {⟨q, q⁻⟩}
16:             queue ← queue ∪ {q⁻}, extended ← true
17:     if extended = false then
18:         R ← R ∪ {⟨q, q⟩}               ▷ stable terminal state
19: Set L(q) = q for all q ∈ Q
20: return M = ⟨Q, I, R, L⟩
```

Exhaustive branching. For each state q, branch on *every* predicate $p(\cdot) \in P_E$ not yet decided in q (line 9), creating successors where $L(q^+) = L(q) \cup \{p\}$ (asserting p true) and $L(q^-) = L(q) \cup \{\neg p\}$ (asserting p false), with transitions $q \to q^+$ and $q \to q^-$. When a state cannot be extended without leading to a contradiction with axioms $\mathcal{A}$, it receives a self-loop (line 21), marking it as a terminal state. A terminal state q is one where for every $p \in P_E$, either $p \in L(q)$ or $\neg p \in L(q)$ (all propositions are decided).

Weakest state and terminal cover. For a KG K, we define the *weakest state* q_K as the state with the smallest labeling (minimal $|L(q)|$) that satisfies all facts in K. Formally, q_K is the state minimizing the number of decided literals while satisfying K:

$$q_K = \arg\min\{|L(q)| : q \in Q, \forall\langle s, p, o\rangle \in K : p(s, o) \in L(q)\}.$$

Terminal cover T_K is the set of terminal states reached from q_K:

$$T_K = \{q \in Q \mid \langle q, q\rangle \in R \wedge q_K \xrightarrow{*} q\}.$$

Axiom-driven pruning. If axioms $\mathcal{A}$ are correct and complete, they can eliminate inconsistent states during construction without threatening coverage guarantees. However, incorrect or incomplete axioms may prune valid worlds, yielding

false coverage guarantees. Expert validation of axioms $\mathcal{A}$ is essential before pruning.

3.2. *Knowledge Graph Validation*

Given a KG K and its Kripke M constructed from the alphabet, validation consists of computing the terminal cover T_K as defined above. The cardinality of T_K determine K's validity:

- $|T_K| = 0$: K is *invalid* (violates axioms $\mathcal{A}$). K's assertions are mutually incompatible given the axioms; axioms pruned states in M that would otherwise model K.
- $|T_K| = 1$: K is *valid and complete* (unique world model). K fully specifies all predicates, yielding a unique interpretation.
- $|T_K| > 1$: K is *incomplete* (multiple consistent interpretations). K underspecifies predicates, leaving multiple consistent valuations.

Each terminal state $q_i \in T_K$ represents a distinct world model satisfying both K's facts and axioms $\mathcal{A}$. Coverage is guaranteed by construction: Algorithm 1 enumerates all feasible truth valuations, ensuring no residual risk arising from undecided predicates is overlooked.

3.3. *Residual Risks Identification*

After establishing validity of KG K (Section 3.2), we assess safety compliance by classifying terminal states in the terminal cover T_K according to safety properties enabling systematic identification of *residual risks*: hazards that remain hidden due to incomplete documentation.

Safety Properties. We classify terminal states using safety properties ϕ expressed as propositional formulas over atomic propositions. A terminal state q is *safe* if $\bigwedge L(q) \models \phi$, and *unsafe* otherwise. These properties can be derived from OWL axioms, SWRL rules, or SHACL constraints following standard practices in semantic web reasoning (Baader, 2003; Heflin et al., 2007).

Realizability. Property ϕ is *realizable* if at least one safe terminal state is reachable in M from initial states; otherwise, ϕ is *unrealizable*.

Let KG K denote the system documentation, with $\varphi(K)$ as its formula representation. Let T_K denote the terminal cover of K (as defined in Section 3). For each terminal state in the terminal cover, we check if its labeling satisfies safety property; i.e., $\forall q_i \in T_K : \bigwedge L(q_i) \models \phi$. The satisfaction patterns reveal distinct risk categories:

Complete and Consistent Documentation. If every terminal state in its terminal cover T_K satisfies safety property ϕ, then KG K is specified completely and consistently.

Unreachable Safety. If no terminal state in its terminal cover T_K satisfies safety property ϕ, then no valuation reachable from K's weakest state q_K is safe. This indicates either that ϕ is unrealizable in Kripke M (no safe terminal state exists anywhere), or that KG K is inherently unsafe (safe terminal states exist in M but are unreachable from weakest state q_K).

Known Unknowns. If some terminal states satisfy the safety property ϕ and others do not, documentation is incomplete, leaving residual risks unaddressed; i.e., $\exists q_s \in T_K : \bigwedge L(q_s) \models \phi \wedge \exists q_h \in T_K : \bigwedge L(q_h) \not\models \phi$. This indicates that KG K is underspecified and does not violate ϕ under OWA. However, under CWA, some reachable worlds violate ϕ, revealing latent hazards.

Risk Severity. For each unsafe terminal state q_h, we measure risk severity as its minimal symmetric difference $|L(q_h) \triangle L(q_s)|$ to any safe terminal state q_s; e.g., how many literal changes are needed to achieve safety. A lower risk severity indicates closer affinity to the designer's intent, while a higher severity indicates many predicate changes are needed, suggesting a systematic safety gap.

4. Automotive Seatbelt Warning System

We present a simplified automotive seatbelt system case study to demonstrate systematic identification of residual risks. This shows how our methodology increases objectivity and decreases reliance on expert judgment in safety analysis.

We deliberately chose this simple, widely understood system to enable readers to follow core contributions and their real-world impact. Despite its simplicity, a seatbelt system contains subtle hazards that were easily overlooked in conventional analysis for decades. The system (restricted to one seat) is required by regulations worldwide (FMVSS 208, ECE R16, Euro NCAP). This sim-

plification is a scope restriction, not an assumption that real systems are simple: we isolate one regulated function to make all modeling assumptions explicit and auditable. The same construction applies to larger systems by expanding the predicate set and partitioning the model into interacting subsystems.

Knowledge Graph. We extract a KG from documentation and observations. Following are the concepts $\mathcal{C}$, relations $\mathcal{R}$, and nodes N.

$$\begin{aligned}\mathcal{C} &= \{\text{Ignition}, \text{Seat}, \text{Seatbelt}, \text{SeatbeltBuckle},\\ &\quad \text{Alarm}, \text{Vehicle}, \text{Person}\}\\ \mathcal{R} &= \{\text{has}, \text{isActive}, \text{isBuckled}, \text{isFastened}, \text{isOn},\\ &\quad \text{isOccupied}, \text{speedExceeds}, \text{speedLimit}\}\\ N &= \{\text{alarm}, \text{driver}, \text{driverSeat}, \text{ignition},\\ &\quad \text{seatbelt}, \text{seatbeltBuckle}, \text{vehicle}\}\end{aligned}$$

The system's documentation specifies:

> *The alarm activates if and only if the ignition is on, the seat is occupied, and the seatbelt is not buckled. When these conditions hold, the speed limiter must engage.*

The formula $\varphi(K)$ of the KG K is:

$$\begin{aligned}\varphi(K) =&[\text{isActive} \iff (\text{isOn} \wedge\\ &\quad \text{isOccupied} \wedge \neg\text{isBuckled})]\\ &\wedge[(\text{isOn} \wedge \text{isOccupied} \wedge \neg\text{isBuckled})\\ &\quad \implies \text{speedLimit}]\,.\end{aligned}$$

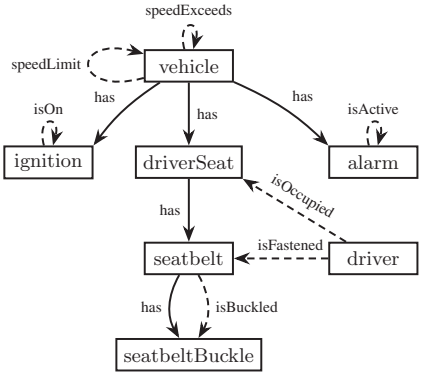

Fig. 1. Simplified seatbelt system (for one seat).

Figure 1 shows the extracted KG of the simplified seatbelt system. Ontological type relationships (e.g., driverSeat is a Seat) are omitted for brevity. Solid arrows represent structural relations (has) and dashed arrows represent mode-dependent relations that can vary across operational modes (e.g., speedExceeds).

Kripke Structure. Node predicates P_N are constant across all states. Edge predicates P_E include both structural (e.g., has) and mode-dependent relationships. We focus on seven binary mode-dependent predicates:

$$\begin{aligned}&\text{isOn}(\text{ignition}, \text{true})\,,\\ &\text{isOccupied}(\text{driverSeat}, \text{true})\,,\\ &\text{isFastened}(\text{driver}, \text{seatbelt})\,,\\ &\text{isBuckled}(\text{seatbelt}, \text{seatbeltBuckle})\,,\\ &\text{isActive}(\text{alarm}, \text{true})\,,\\ &\text{speedExceeds}(\text{vehicle}, 20\text{kmh})\,,\\ &\text{speedLimit}(\text{vehicle}, 20\text{kmh})\end{aligned}$$

For brevity, we omit explicit arguments of predicates and refer to $\text{isFastened}(\text{driver}, \text{seatbelt})$ simply as isFastened, and so forth.

Considerations. We distinguish sensor readings (e.g., speedExceeds) from actuator outputs (e.g., speedLimit). We also assume structural completeness: a seatbelt warning system possesses all structural components. While our methodology supports verifying structural completeness, here we only focus on validating and verifying behavioral completeness and correctness, which present greater analytical challenges and require substantial expert effort.

Ontological Axioms. Four axioms constrain the state space. The core axiom reflects reality:

A1. *If a person is properly wearing the seatbelt for restraint, then the buckle must be latched.*

$$\text{isFastened} \implies \text{isBuckled}\,.$$

Three additional axioms are optionally enforced depending on system assumptions:

A2. *Belt cannot be fastened with no occupant:*

$$\neg\text{isOccupied} \implies \neg\text{isFastened}\,.$$

A3. *Buckle cannot be latched with no occupant:*

$$\neg\text{isOccupied} \implies \neg\text{isBuckled}\,.$$

A4. *Speed cannot exceed 20 km/h if ignition off:*

$$\neg\text{isOn} \implies \neg\text{speedExceeds}\,.$$

Table 1 shows axiom-driven pruning. With 7 binary mode-dependent predicates, the theoretical

maximum is 128 terminals. Each axiom combination prunes infeasible states, with the number of unsafe terminals varying depending on which operational contexts are deemed feasible.

4.1. *Latent Hazard Detection Framework*

To identify residual risks, we compute the terminal cover T_K of the extracted KG; i.e., terminal states where $q_i \in T_K : \bigwedge L(q_i) \models \varphi(K)$. Of note, the documented system is supposedly safe.

Safety Properties. We specify two safety properties capturing possible latent hazards:

$$\phi_{\text{fasten}} = \text{isBuckled} \wedge \neg\text{isFastened}$$

$$\phi_{\text{speed}} = \text{speedLimit} \wedge \text{speedExceeds}$$

We classify states into three tiers: *documented safety* (terminal states that satisfy $\varphi(K)$), *misuse hazard* (terminal states that satisfy ϕ_{fasten}), and *enforcement hazard* (terminal states that satisfy ϕ_{speed}). A state is *truly safe* iff it satisfies $\varphi(K)$ and satisfies neither latent hazard property.

With all four axioms enabled (A1–A4), we obtain 48 terminal states: 22 satisfy $\varphi(K)$ and 26 are unsafe. Table 2 shows that fewer than 60% of documented-safe states are truly safe, with the remainder harboring latent hazards. We now examine each hazard type in detail.

Table 1. State space size by axiom-driven pruning.

#	Axioms	$\lvert Q\rvert$	Safe	Unsafe	Pruned
1	None	128	60	68	0
2	A1	96	46	50	32
3	A2	96	44	52	32
4	A3	96	44	52	32
5	A4	96	44	52	32
6	A1+A2	80	38	42	48
7	A1+A3	64	30	34	64
8	A1+A4	72	34	38	56
9	All (A1–A4)	48	22	26	80

Table 2. Latent hazard analysis with all axioms enabled.

Classification	Count	% of Safe
States satisfying $\varphi(K)$	22	100%
Truly safe (no latent hazards)	13	59.1%
Misuse hazard (ϕ_{fasten})	5	22.7%
Enforcement hazard (ϕ_{speed})	3	13.6%
Both latent hazards	1	4.5%
Coverage gap	**9**	**40.9%**

4.2. *Misuse Hazards*

Documentation conflates person restrained (i.e., isFastened) and buckle sensor triggered (i.e., isBuckled), implicitly assuming:

$$\text{isFastened} \iff \text{isBuckled}\,.$$

Realistically, designs only enforce:

$$\text{isFastened} \implies \text{isBuckled}\,.$$

The reverse fails: occupants can buckle behind their back or under their arm without proper restraint. This creates a misuse hazard where the system observes $\text{isBuckled} = \mathsf{true}$ but the actual safety state is $\text{isFastened} = \mathsf{false}$.

Five states satisfy $\varphi(K)$ but satisfy ϕ_{fasten}. The system believes these states are safe (buckle latched, alarm off, speed unrestricted), yet the occupant lacks crash protection. The $\varphi(K)$ uses only isBuckled because documentation can only reference measurable system states, but this leaves the misuse case undetected.

4.3. *Enforcement Hazards*

Documentation states *"speed is limited to 20 km/h when unfastened"* without distinguishing the control action (i.e., speedLimit) from the observation (i.e., speedExceeds). While $\varphi(K)$ correctly uses speedLimit, the specification assumes $\text{speedExceeds} = \mathsf{false}$ if $\text{speedLimit} = \mathsf{true}$.

In low-speed scenarios (15 km/h with unbuckled belt), engaging the speed limiter safely prevents acceleration beyond 20 km/h. However, in high-speed scenarios (120 km/h on highway when belt becomes unbuckled), the same control action creates a critical hazard: abrupt deceleration causes loss of control and multi-vehicle collisions (Hu et al., 2016; Soica and Gheorghe, 2025).

Three states satisfy $\varphi(K)$ but satisfy ϕ_{speed}: the speed limiter is engaged ($\text{speedLimit} = \mathsf{true}$) while the vehicle travels at high speed ($\text{speedExceeds} = \mathsf{true}$). This represents a *temporal hazard*: the state itself is the dangerous transition where active enforcement causes abrupt deceleration. Eventually speedExceeds should become false, but the path to compliance is hazardous. Our methodology exposes this by treating specifications as properties to verify, not requirements to implement (Rushby, 2009).

4.4. *Summary and Implications*

Beyond the two individual hazard types, one state exhibits both misuse and enforcement hazards simultaneously: belt buckled at high speed and the system attempts deceleration enforcing safe speed limits. This compounds risks, as the occupant is unrestrained during dangerous braking.

This coverage gap demonstrates that nearly half of specification-compliant states harbor undocumented hazards. Traditional validation checking only $\varphi(K)$ would approve these states, missing latent risks that arise from conflating observables with unobservables and control outputs with sensor inputs. Our exhaustive enumeration provides formal assurance that all hazards within the modeled scope are identified.

5. Related Work

OWA reasoners check consistency but do not identify latent hazards in incomplete specifications or distinguish false negatives (known unknowns) from false positives (contextual ambiguities) (Baader, 2003; Heflin et al., 2007; Knublauch and Kontokostas, 2017). Gueffaz et al. (2012) translate RDF to Kripke structures by treating states as nodes for navigation, not as truth valuations. They neither branch to materialize known unknowns, nor expose contextual ambiguities.

Hazard analysis rely on checklists and expert judgment, making it fundamentally incomplete and error-prone as they depend on analysts imagining scenarios (Leveson, 2012). We mechanized risk identification by enumerating possible valuations and transforming implicit residual risks into explicit findings with guarantees.

Temporal RDF extensions address temporal modalities but lack safety analysis capabilities (Hartig and Thompson, 2014). Our contribution bridges ontological KGs with modal CWA reasoning, inheriting mature formal methods tooling while respecting OWA flexibility. Meanwhile, model checking verifies properties over complete models under CWA (Clarke et al., 2018). This work complements CWA by extending it to incomplete models under OWA through construction-time reasoning.

Embedding-based KG completion (Ji et al., 2022) predicts missing triples statistically but cannot distinguish documented from latent safety requirements. Our contribution explicitly models this distinction through three-tier classification (documented safety, misuse hazards, enforcement hazards), enabling coverage gap quantification that embedding methods cannot provide. Statistical approaches optimize for prediction accuracy on historical data; our exhaustive enumeration guarantees coverage over all feasible states within the declared alphabet.

6. Conclusion

We presented Kripke semantics with exhaustive branching that guarantees coverage over all truth valuations, together with three identification patterns that discovers latent hazards (Known Unknowns). The construction-time validation treats contradictory documentation as distinct world models rather than errors requiring resolution upfront, reducing reliance on expert judgment.

The case study demonstrates that the gap between OWA and CWA manifests in real systems: documentation exhibits critical conflations. First, it conflates predicates like $\mathrm{isFastened}$ and $\mathrm{isBuckled}$ that appear synonymous but carry different safety implications (observable sensor state vs. unobservable proper restraint). Second, it conflates $\mathrm{speedLimit}$ (control output) and $\mathrm{speedExceeds}$ (sensor input), creating enforcement hazards where engaging the speed limiter at highway speeds causes dangerous abrupt deceleration. Our methodology exposed these hazards through exhaustive enumeration: with 7 predicates generating 128 possible states (48 feasible under axioms), we identified 22 safe and 26 unsafe states. Crucially, 40.9% of states marked safe by documented specifications harbor latent hazards, demonstrating systematic underspecification.

Our framework distinguishes *documented safety* (satisfaction of explicit requirements) from *actual safety* (absence of all hazards including latent ones). This distinction is critical for:

- **Accountability:** When hazards manifest, was the state documented-safe but actually-unsafe (specification gap) or documented-unsafe (implementation failure)?

- **Completeness:** Coverage gap quantifies specification quality
- **Traceability:** Latent hazards trace to missing documentation, not faulty implementation

By exposing known unknowns as explicit terminals, the framework enables precise attribution when hazards occur: the Kripke structure reveals whether risks were enumerated and accepted, or genuinely unforeseen. This distinction matters for accountability where liability depends on demonstrating due diligence. Identifying contextual ambiguities prevents engineers from implementing specifications that are safe in one context but hazardous in another, which is a class of errors that has caused incidents across various domains.

Like many formal verification methods that provide correctness guarantees, our approach has exponential worst-case time and memory complexity in the number of mode-dependent predicates, with $|P_E|$ as the exponent. This is the cost of exhaustive coverage. In practice, this can be alleviated through compositional decomposition, symbolic representations (e.g., SAT/SMT or BDD-based encodings), and property-guided reduction techniques; we defer a full scalability treatment and evaluation to future work.

Regarding external validity, the seatbelt study must be read as a controlled proof-of-concept demonstrating failure modes the method detects. Application to real-world is feasible when system documentation is already structured into requirements, hazards, and trace links; in that setting, the method scales by modular analysis and staged integration rather than monolithic modeling.

Extending this work requires addressing several challenges: handling unknown unknowns through iterative ontology refinement beyond closed alphabet assumptions; scaling to industrial systems via partial order reduction, symbolic representations, and incremental validation; developing automated repair strategies using delta debugging (Zeller, 1999; Ebrahimi et al., 2026) to transform unsafe terminals into safe ones.

Acknowledgement

This work was supported by the ASSURE project (MDU multi-disciplinary research initiative) and in part by the ITEA4 24026 CLEAR (Comprehensive Learning for Enhanced AI Responsiveness) project.

References

Alur, R. (2015). *Principles of Cyber-Physical Systems*. Cambridge, MA, USA: The MIT Press.

Baader, F. (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge University Press.

Clarke, E. M., O. Grumberg, D. Kroening, D. A. Peled, and H. Veith (2018). *Model Checking* (2nd ed.). Cambridge, MA, USA: MIT Press.

Ebrahimi, M., K. Hänninen, K. Lundqvist, and M. Sirjani (2026). When Repair is not Enough: Systematic Mitigation Strategies for Incomplete Knowledge Graphs. Submitted to ESREL 2026. Companion paper presenting specification-level mitigation framework.

Ehrlinger, L. and W. Wöß (2016). Towards a definition of knowledge graphs. *SEMANTiCS 48*(1-4), 2.

Gruber, T. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition 5*(2), 199–220.

Gueffaz, M., S. Rampacek, and C. Nicolle (2012). Temporal logic to query semantic graphs using the model checking method. *J. Softw. 7*(7), 1462–1472.

Hall, A. T., D. D. Frink, and M. R. Buckley (2017). An accountability account: A review and synthesis of the theoretical and empirical research on felt accountability. *Journal of Organizational Behavior 38*(2), 204–224.

Hartig, O. and B. Thompson (2014). Foundations of an alternative approach to reification in rdf. *CoRR abs/1406.3399*.

Heflin, J. et al. (2007). An introduction to the owl web ontology language. *Lehigh University. National Science Foundation (NSF) 7*.

Hu, J., J. D. Rupp, K. Fischer, P. Lange, and A. Adler (2016). Optimizing seat belt and airbag designs for rear seat occupant protection in frontal crashes. In *Proceedings of the IRCOBI Conference 2016*, pp. 67–100.

ISO (2018). BS EN ISO 9241-304:2008: Ergonomics of human-system interaction. ISO standard.

Ji, S., S. Pan, E. Cambria, P. Marttinen, and P. S. Yu (2022). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems 33*(2), 494–514.

Knublauch, H. and D. Kontokostas (2017, July). Shapes constraint language (shacl). W3c recommendation, W3C.

Leveson, N. G. (2012). *Engineering a Safer World: Systems Thinking Applied to Safety*. MIT Press.

Musen, M. A. (2015). The protégé project: a look back and a look forward. In *AI Matters*, Volume 1, pp. 4–12. ACM.

Palshikar, G. K. (2001). Applying formal specifications to real-world software development. *IEEE Softw. 18*(6), 89–97.

Rushby, J. (2009). Software verification and system assurance. In *SEFM 2009*, pp. 3–10. IEEE Computer Society.

Soica, A. and C. Gheorghe (2025). A review of seatbelt technologies and their role in vehicle safety. *Applied Sciences 15*(10).

Zeller, A. (1999). Yesterday, my program worked. today, it does not. why? In *Software Engineering — ESEC/FSE '99*, Volume 1687 of *Lecture Notes in Computer Science*, pp. 253–267. Springer.