Defining a Minimal Set of Trustworthy Properties for Reliable Knowledge-Based Systems

Florence de Grancey* Gaëlle Lortal† Claire Laudy†
Amandine Audouy* Florent Chenevier* Joshua Salort*
*Thales, France
†cortAIx Labs, Thales, France
†email:firstname.lastname@thalesgroup.com

***Abstract*—Knowledge-Based Components (KBCs) are a promising technology for a high-risk Artificial Intelligence (AI) system. Nevertheless, ensuring their reliability becomes challenging as Knowledge Models and reasoning mechanisms grow in complexity. Existing trustworthy-AI frameworks predominantly focus on learning systems and offer limited guidance for symbolic of knowledge driven systems. As an initial step toward filling this gap, we propose a minimal set of trustworthy properties specifically tailored to KBCs. These properties are derived from a state-of-the-art analysis, a risk-based examination of potential failure modes, and an adaptation of established trustworthiness principles from the learning domain. We analyze how these properties influence the development process of KBCs and illustrate their practical relevance through a supportive example.**

Keywords-Artificial Intelligence; Knowledge Processing; Knowledge Representation Formalisms and Methods; Knowledge base verification.

I. Introduction

While Artificial Intelligence (AI) is now widely deployed for every-day life applications, its integration into high risk systems, such as nuclear plants or aircraft, remains a major challenge. Such systems are characterized by the possibility of fatal consequences in case of failure, including human injury or environmental damage. They are therefore developed under a stringent and reliable development process and strong regulatory constraints. In the context of AI, *trustworthy AI engineering* practices are essential to guarantee AI system performance, safety and robustness, and its compliance to emerging regulation, such as the AI Act [1]. Theses practices are build upon established classical engineering methods practices (such requirement writing and verification, configuration management) ensuring system reliability, and extended through the definition and verification of *trustworthy properties*. Theses properties may apply in the scope of the AI system or in the scope of items, such as datasets or trained models in the case of supervised learning technologies.

To address the challenge of AI based high-risk applications, the industry is also exploring alternative AI technologies, such as *Symbolic AI* (SAI). Symbolic approaches rely on the formalization of knowledge and facts (general, domain-specific, expert) using logics, and a combination with reasoning algorithm to infer new knowledge. This family includes expert-systems, ontology-based reasoning and constraint problem solving. SAI offers attractive features for high-risk systems, such as explainability by design and reasoning algorithm correctness.

Nevertheless, when applications grows in complexity, requiring large Knowledge Models and complex reasoning algorithms, AI engineer may no longer be able to anticipate the system outcome. Developers may be unable to design the exhaustive required test suite covering the combinatorial explosion of possible facts, knowledge, and reasoning situations. In this context, identification of trustworthy properties for KBC becomes a promising approach to ensure reliability.

Extending the methodological works on ontology building [2] in a water resources monitoring project [1], our contribution provides a starting point toward defining a minimal set of trustworthy properties for SAI System. Building on the state of the art in trustworthy properties for AI, completed by a risk based approach, we identified and defined a preliminary set of properties. Their impact on engineering process is also analyzed. The paper describes the analysis performed to identify trustworthy properties in section 2. Section 3 describes the selected set of properties and their applicability in the context of knowledge base system development. Section 4 presents an illustrative example of theses properties. A discussion is provided in the last section.

II. Defining trustworthy properties

A. Knowledge based component

To support our analysis, we first define the scope under consideration. We consider an AI-based system that incorporates a software component developed using SAI technology, referred as the *Knowledge Based Component* (KBC). A KBC can generally be structured as a combination of several elements, as illustrated in Figure 1.

- A Knowledge Model, which contains the formalized knowledge. It includes general concepts (e.g., "cars", "road",..) and contextual information (e.g., "N124 is a road","N124 is open"). The knowledge is expressed in a dedicated knowledge representation language, defining the allowed syntactic and semantic rules.
- A reasoning engine, which applies a reasoning algorithm to infer new knowledge from the existing knowledge model.

[1]This research was funded, by the Agence Nationale de la Recherche (ANR) through TETRA Project, grant ANR- 22-FAI2-0006-01

Algorithm rely on a set of logical rules, iteratively applied to each element of the knowledge model.
- A Request engine (or query engine), which enables interaction with the knowledge model, by retrieving relevant information. It usually relies on specific search algorithms designed for efficiency.

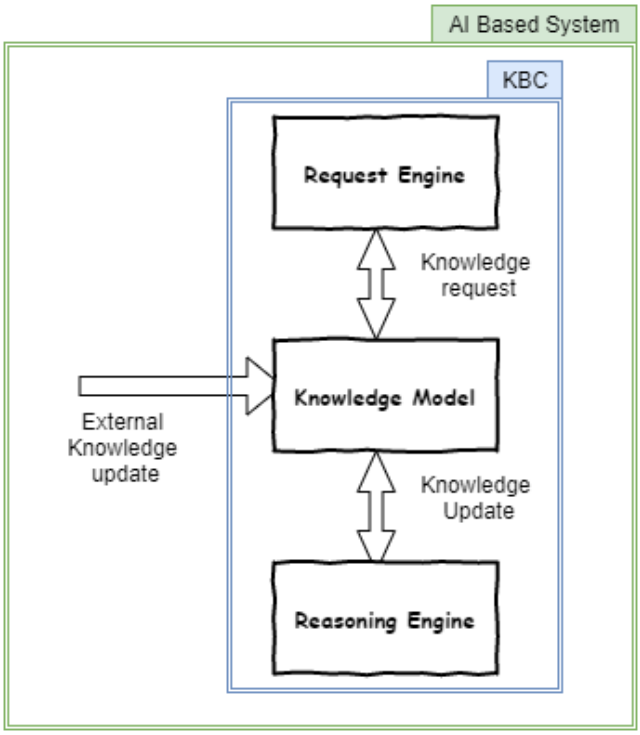

Figure 1. Knowledge Based Component schematic architecture.

It is important to note that the functional behavior emerges from the KBC as a whole, rather than individual elements: A knowledge model alone is not usable without a query mechanism to access its content. Although, this decomposition is inspired by expert systems, it can be generalized to other SAI technologies such constraint-solving problems. In this case, the set of variable, domains, and constraint constitute the knowledge model, while the constraint solver acts as the reasoning engine.

B. Methodology

To define the trustworthy properties for reliable KBC, we apply an exploration methodology structured along three complementary paths:
- State of the art analysis, focusing on trustworthy properties relevant to the context of knowledge modeling and symbolic reasoning.
- Bottom-Up analysis starting from potential errors and risks associated with KBCs and identifying the trustworthy properties required to mitigate them.
- A top-down approach, examining trustworthy properties established in the machine-learning domain and assessing their applicability to KBCs.

C. Path 1: State of the art analysis

Knowledge-Base engineering has provided standards[2], methodologies [3], [4], [5], [6], [7], and tools [8], [9], to build, use and maintain KB and thus provide confidence in KBS. Recent works are more often focused on specific technologies.

Ontology engineering One of the most influential contributions comes from the 2013 Ontology Summit [10], which proposed an eight-step iterative development cycle. This development cycle is driven by five properties: *Intelligibility* ("Can humans understand the ontology correctly?"), *fidelity* ("Does the ontology accurately represent its domain?"), *Craftsmanship* ("Is the ontology well-built and are design decisions followed consistently?"), *Fitness* ("Does the representation of the domain fit the requirements for its intended use?) and *Deployability* ("Does the deployed ontology meet the requirements of the information system of which it is part?"). Additional designed-oriented properties are described in [11] including *Clarity*, *Coherence*, *Extendibility* ("An ontology should be designed to anticipate the uses of the shared vocabulary."), *Minimal encoding bias*, *Minimal ontological commitment*. While theses properties capture relevant desirable qualities for a KBCs, they lack clear definitions and associated metrics or measurement methods.

Knowledge graphs: Knowledge Graphs (KGs) are widely used to store, represent, and manage structured information about entities and their relation. Assessing and evaluating their quality has become an active research axes. [12] provided a comprehensive review of quality requirements in the context of KGs. [13] address the challenge by proposing a set of properties and tractable metrics, such as *"Accuracy and Correctness to measure the degree to which the KG reflects ground truth"* or *"Completeness measures how thoroughly the Knowledge Graph covers its intended domain"*. Generalization of theses metrics to KBCs can be considered;

Neuro-Symbolic AI: Neuro-Symbolic AI (NSAI) is an active research domain aiming at integrating symbolic reasoning and deep-learning, to leverage qualities of both technologies. As detailed in this comprehensive survey, [14], NSAI enforces by design interpretability, robustness and fairness, which are desirable properties for a learning system. However, [15] pinpoint that the development of trustworthiness methods is largely driven by application needs and lack of a general framework. The work of [16] assesses the application of AI Act desirable features for an AI System (e.g., *Human oversight*, *robustness*, *fairness*), in the context of NSAI and shows that a plurality of metrics is still missing to perform the complete evaluation.

D. Path 2: A Risk-Based Analysis

To ensure the reliability of a KBC deployed in high-risk system, the development process must minimize the occurrence of runtime errors. Such errors may affect the system in two ways:

[2]https://www.w3.org/standards/ Accessed: Mar. 5, 2026.

- **Reduced availability**: the component fails to deliver its intended function due to a detected error or a component loss.
- **Reduced integrity**: the component provides an erroneous output, that remains undetected.

Ensuring that a set of trustworthy properties is hold throughout development and operation is a promising approach to mitigation. So, we conduct a systematic analysis of KBC and its components potential errors, and linked to a specific risk. We then derive the adequate trustworthy property required for mitigation.

Knowledge Representation Language (KRL): We assume that the KRL is selected at design time and remains fixed during KBC operations. Errors occur exclusively at design-time and may take the following forms:

- The choice of an over-expressive KRL: High expressivity language can support complex constructs that simplify the modeling (such as the "existential" quantifier). However, it may lead to undecidability (e.g., rule application does not guarantee termination) or to prohibitive computational costs (e.g., exponential time reasoning). Such design choice can lead to an *availability* risk.
- Conversely, selecting a KRL with insufficient expressive power can lead to oversimplification of the Knowledge Model and a potential *integrity* risk.

Mitigation by property 1: We can define a **Language Suitability** property, defined as "the extend to which the knowledge representation language contains only syntactic elements or rules required for considered use cases". This property is related to *Fitness* described in [10].

Knowledge modeling: Knowledge is formalized through the association of general concepts ("car", "road") and instantiated in instance-level elements representing real-world entities ("TN-822" as car number). Errors in knowledge modeling mainly arises from semantically incorrect content, such as false knowledge (“cars can fly”) or inconsistent knowledge (both “all cars have only four wheels” and “all cars have only two wheels”). Such errors directly lead to an *integrity risk*.
Mitigation by property 2: We define a **knowledge correctness** property as the semantic accuracy of the knowledge contained in the KBC. It is worthy to note that it can be refined into a *knowledge consistency* e.g., the Knowledge Model does not contain contradictory elements.

Even when semantic correctness is ensured, additional modeling errors may occur:

- Missing knowledge: required elements lack for executing a use case. It leads to errors and *availability* risks.
- Excess knowledge: irrelevant elements with respect to the UC, increasing the size and complexity. It degrades reasoning performance and leads to *availability* risks.

Mitigation by property 3: We define a **Knowledge Suitability property**, i.e. "the Knowledge Model contains exactly the necessary knowledge element for the use-case". "Necessary knowledge elements" are used at runtime.

Reasoning Algorithm: Reasoning algorithms infer new facts from the existing Knowledge Model. They are evaluated by their *soundness* (all the derived inference are true from semantic point of view), their *completeness* (an algorithm is complete if, when deriving formulas, all the formulas are well derived), and their *algorithmic complexity*. Theses properties allow to identify errors related to:

- Non-sound algorithm e.g., some derived inference may not be true, leading to *integrity risk*.
- Incomplete algorithm e.g., some inference are not computed, leading to erroneous output and *integrity risk*.
- Algorithm that has no termination in a reasonable time for some Knowledge Models, leading to *availability risk*.

Mitigation by property 4: We define the **Algorithm suitability** property, stating "that the reasoning algorithm and its constraints (response times, semantic accuracy of responses) are appropriate for the selected use case". This property is close to *Craftsmanship* in [10].

Request Algorithm: They exhibit similar properties, errors and risk as an information search algorithm, e.g., recall completeness (all requested elements are provided), precision (too much elements are provided) or non-acceptable termination time.

Mitigation by property 5: We extend the suitability concept to define a **Request Algorithm suitability** ensuring that "the request mechanism is adapted to the use case and its performance constraints".

E. Path 3: Extension of Machine Learning trustworthy properties

The third path to define trustworthy properties consists in examining how properties established for learning-based AI systems can be extended to Knowledge-Based Components. Trustworthy AI properties for machine learning have been extensively studied in the literature, notably in the DEEL project white paper [17] and the confiance.ai report [18]. From these analysis, several categories of properties can be identified:

- Properties related to desirable engineering feature. Such properties are independent of the underlying (AI) technology. Examples are “Auditability”, ”Maintenability”, “Resilience”, “Specifiability” and “Verifiability”. Theses properties can be directly transferred to KBCs development process, without modifications.
- Properties can can be easily adapted to KBCs, assuming the specificity of knowledge based applications. For instance, the **"Data Quality"** defined in [17] as *"the extent to which data are free of defects and possess the desired features"* can be transposed to a Knowledge Quality property, aligned with the notions introduced in the risk-based analysis. Similarly, **Explainability**, defined as *“the extent to which the behavior of a ML model can be made*, can be applied to KBCs by replacing the notion of “ML model” with the knowledge base and its associated reasoning mechanisms. It is close to the explainability concepts explored in [15].

III. Minimal set of trustworthy properties

A. Properties selection

Grounded in the three exploration paths, we select a minimal set of trustworthy properties based on three criteria:

- If the property mitigates a risk when satisfied ?
- If it is possible to verify the property at least once during KBC development process (see related section) ?
- If the selected properties are consistent, non-redundant with another selected property ?

We obtain the set of properties given in Table I. It is worth noting that we define both properties related to the whole component, as it hosts the functional behavior of the component, and properties related to KBC elements, as they drive specific failures.

In addition to this set, we advocate that **KBC explainability** should also be considered. The *Development explainability* could be a useful method to detect unexpected behaviors during development. *Operational explainability*, or explainability for the end-user may be a crucial property for acceptability and trust. In the context of KBCs, explainability is related to the ability to store or reconstruct the inference or query path within the Knowledge Model ([19]).

TABLE I. Proposed minimal set of Trustworthy properties for a KBC (A= Availability risk, I = Integrity risk).

Property	Definition	Mitigate
Language suitability	The extend to which the chosen knowledge representation language contains only the syntactic constructs required for the use case	A,I
Knowledge correctness	The extend to which the Knowledge Model is logically and semantically consistent within the use case needs	I
Knowledge completeness	The extend to which the Knowledge Model contains the necessary knowledge to satisfy use case needs	I
Knowledge suitability	The extend to which the Knowledge Model contains the necessary knowledge to satisfy use case needs	A,I
Algorithm suitability	The extend to which the selected reasoning algorithm fits to the use case needs and its constraints	A,I
KBC performance	The extend to which the selected reasoning algorithm fits to the use case performance needs	A,I
KBC stability	The extend to which the KBC fits to the use case stability constraints	A,I
KBC robustness	The extend to which the KBC is able to maintain its behavior in adverse conditions	A,I

B. Verifiability and impact on the engineering process

The defined Trustworthy properties can be smoothly integrated into the development process of a system containing a Knowledge Base Component, as illustrated in Figure 2.

This development cycle begins with traditional system-engineering activities, including the elicitation of end-user needs, the definition of Concepts of Operation, and the specification of system requirements and architecture. A particular attention should be paid to define a complete and representative set of operational scenarios describing how the system will be used. These scenarios will later guide the construction of test cases at the KBC level.

Once responsibilities and requirements have been allocated to the KBC, the proper development of the KBC begins. Following the approach proposed in the confiance.ai project [20], we recommend to consider a component development cycle divided into two main steps:

- A functional Design and Verification/Validation step, which aims to develop the KBC and its internal elements according to the allocated requirements, then to verify the correctness of its functional behavior. For clarity, we distinguish a **functional design phase** from a **functional (validation and) verification** phase.
- An implementation step, where the designed KBC is concretely implemented in the appropriate software stack (e.g., programming language, operating system) and in the appropriate hardware platform (e.g., CPU, FPGA).

We now detail how trustworthy properties are integrated into theses development steps.

Functional design phase: This phase includes activities related to knowledge modeling, selection of the appropriate knowledge-representation language, and development of the reasoning and querying algorithm. Theses activities are typically carried out iteratively to converge toward the most suitable design choice. Trustworthy properties can be introduced at this stage as desirable qualities, guiding the design or expressed directly as requirements. For example, *language suitability* may be used as a design constraint and lead to a requirement, such as *The Knowledge Model shall be expressed in OWL-EL profile* for an ontology-based application. The property *knowledge correctness*, may lead to define supplementary requirements related to runtime validation rules, such as SHACL rules [21] for an ontology-based application. The property *algorithm suitability* may motivated requirements related to the soundness and completeness of the selected reasoning or querying algorithms.

During knowledge modeling, we encourage to set-up a **bidirectional traceability** between the operational scenarios defined at system level and knowledge elements contained in Knowledge Model. It can be an effective method to ensure *knowledge completeness* and *knowledge suitability* during the design.

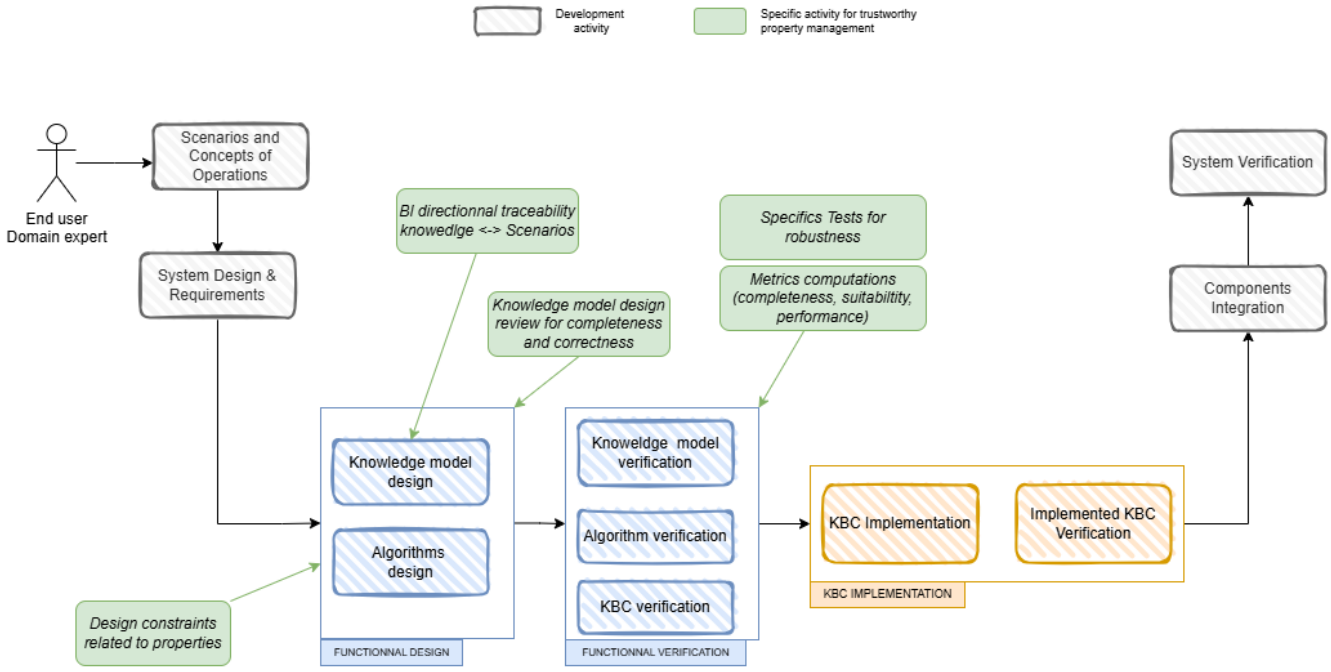

Figure 2. (Simplified) Knowledge Based Component development process.

At the end of the design process, we also recommend to conduct **knowledge-model design review** that can be performed to assess properties knowledge related properties. This review can be assisted by tools to compute metrics on the Knowledge Model, such as OntoMetrics in the ontology context ([22]). Custom metrics may be defined, such as syntactic construct coverage metric as the number of syntactic construct used over the total number of syntactic constructs of the KRL, for *language suitability*.

Functional verification of the KBC: This step focuses on validating and verifying the designed KBC with respect to its requirements and trustworthy properties. The usual practice is to build tests procedures and tests cases that exercise the KBC and its elements under various querying and reasoning conditions. The set of operational scenarios identified during system-level activities may be used as guidance during their elaboration. Several trustworthy properties can be assessed through this scenario-based evaluation: *Algorithm suitability* can be verified using measurement of computational load and the accuracy of the KBC outcome; *Knowledge suitability* can be assessed by computing a knowledge coverage metric defined as the number of knowledge elements used over the total number of knowledge elements. Complementary tests can be developed to address specific trustworthy properties like *KBC robustness*. This property will require evaluating the KBC under particular configurations of knowledge elements or atypical request patterns, that may be tool-assisted [23].

KBC implementation Once the functional behavior of the designed KBC has been validated, the proper **implementation** activities can begin. This phase consists of implementing the KBC on the appropriate software stack and/or hardware platform. To the best of our knowledge, traditional implementation activities like software development are directly applicable. After implementation, compliance with the trustworthy properties should then be re-verified on the implemented KBC, to guarantee that no degradations occurs during the process. Only after this verification, the implemented KBC be integrated into the broader system-engineering workflow. The operational scenarios defined during system design can then be reused to confirm that the final system continues to meet end-user needs.

C. Use case illustration

We illustrate the proposed trustworthy properties using a simple supportive example. Consider a use case in the automotive domain, where the system assists the driver by answering drivers questions, such as *if their car can drive on a specific road?*. A system embedding Knowledge Based component is used to compute reliable answers.

The designed Knowledge Model (KM) is represented in Figure 3. We adopt the following notation: in square brackets or blue in the figure, the general concepts (ex: [car]), in brackets or orange in the figure, the facts (ex: (TN-822) as identifier of the driver's car). The KM also contains a simple deductive rule R1:

```
(R1) A [Car] CanDriveOn [Road] if [Road]
isTypeOf (Highway) or (Back road)
```

We define two operational scenario related to queries (Q):

```
Q1: Is (TN-822) CanDriveOn (A64) ?
Expected Outcome: Yes
```

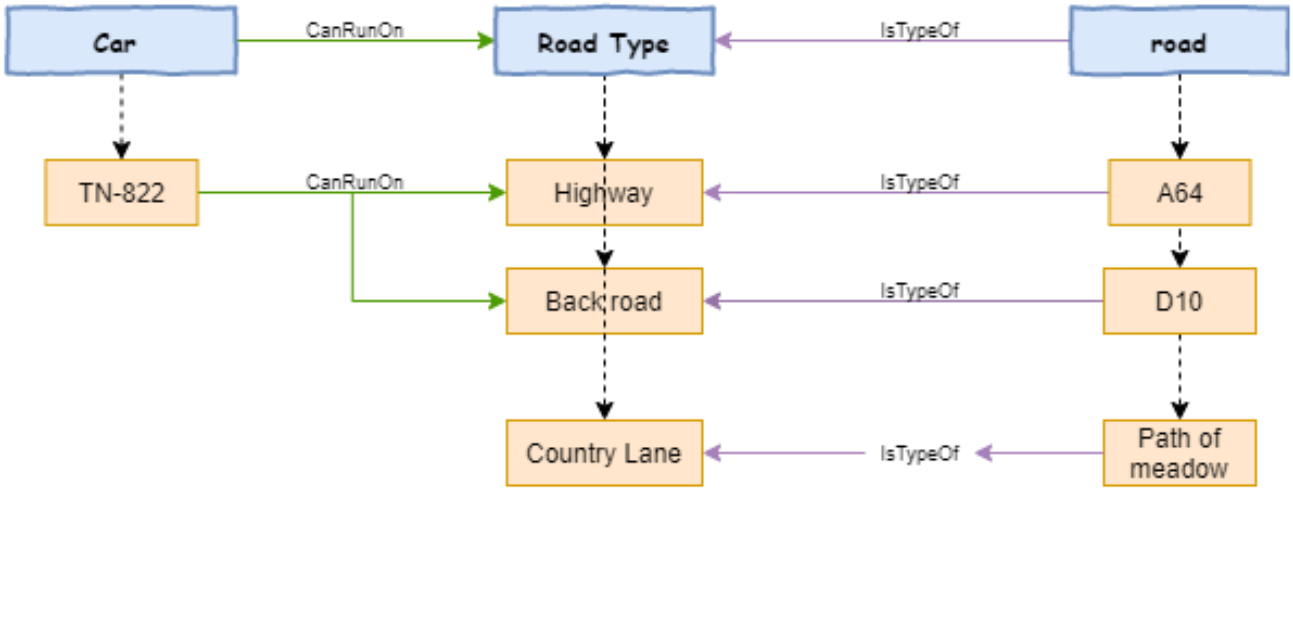

Figure 3. Automotive Use Case Knowledge Model. Concepts are represented in blue rectangles, facts in oranges rectangles. Relations between concepts are arrows with text.

```
-----
Q2: On which [road] (TN-822)
CanDriveOn ?
Expected Outcome: A64,D10
```

Is it worth noting that the KM does not explicitly express the expected outcomes in both scenarios. The answers are obtained by applying the deductive rule to the KM.

We now illustrate potential errors are mitigated by trustworthy properties:

Knowledge correctness: This property is violated if an element is incorrectly specified in the KM. For example, if the road (A64) is assigned an incorrect type, the query Q1 produces an erroneous outcome:

```
KM error: "(A64) isTypeOf (CountryLane)"
Q1: Is (TN-822) CanDriveOn (A64) ?
Outcome: No
```

Knowledge completeness: This property is violated if a necessary knowledge element is missing. In such a case, the KBC cannot produce the correct answer:

```
KM error: Missing Element
"(A64) isTypeOf (Highway)"
Q2: On which [road] (TN-822) CanDriveOn ?
Expected Outcome: D10
```

Knowledge suitability: The property will be violated if the KM contains concepts of [bike] "canRunOn" [CountryLane], which is not necessary for the use case. The outcome is still true but the reasoning time can be increased.

```
Knowledge base additionnal element:
[bike]  "canRunOn" [CountryLane]
Q1:  Is (TN-822) CanDriveOn (A64) ?
Outcome: Yes
```

Algorithm suitability: The property is not satisfied in this use case if the selected algorithm is unable to apply the deductive rule. In such a situation, the KBC can no longer produce the expected result.

KBC stability: Stability can be assessed by testing different formalization of the request, such as expressed below. A stable KBC will answer yes regardless the formalization.

```
Q1a: Is my (TN-822) CanDriveOn (A64) ?
Q1b: Is (TN-822) CanDriveOn (A64)
or not?
Expected Outcome: Yes
```

KBC robustness: On the toy use case, KBC robustness can be experimented by assuming errors on the query, such as:

```
Q3a: Is (TN-821) CanDriveOn (A64) ?
Expected Outcome: No
Q3b: Is (TN-822) CanDriveOn (D11) ?
Expected Outcome: No
```

IV. Discussion

Based on the state-of-the art analysis and the risk-based approach, we propose a minimal set of trustworthy properties for KBCs. We acknowledge that this minimal set was developed from an engineering-oriented perspective and could be enriched with additional properties addressing human-factors or ethical considerations. For example, particular attention should be paid to undesirable biases, which may easily emerge during knowledge modeling and lead to unappropriate use of the system. A property of **Knowledge fairness** may be defined as Knowledge Model level to mitigate this ethical risk, and verified during the knowledge design review.

A mandatory criterion for selecting the properties in this minimal set was the ability to **verify** them during development process. We do not prescribe any specific verification methodology - design review, test, traceability analysis,... -, however, we pinpoint that a formal demonstration of each property is currently out of reach due to the absence of a sufficiently mature mathematical framework. We therefore encourage the academic community to pursue the formalization of these properties and the development of associated verification methods. This would constitute a valuable contribution to improving the reliability of knowledge-based applications for high-risk systems.

The relevance of trustworthy properties approach may be challenged by considerations related to use case complexity. In simple rule-based systems, where the behavior can be fully specified and the reasoning/query space can be exhaustively validated using a manageable set of test scenarios, such properties may appear unnecessary. In these situations, traditional verification techniques are often sufficient to guaranty the reliability of the KBC. We argue that trustworthy properties become essential once the system exceeds a certain **complexity threshold**. This threshold can be characterized by several criteria, including:

- the domain expert's ability to handle and understand the KM (without being a knowledge modeling expert).
- the number of reasoning steps required to answer user needs.
- the developer's ability to build the adequate and exhaustive set of scenarios exercising the reasoning and query over the KBC,
- the domain expert's ability to anticipate the KBC's outcomes on these scenarios.

Such criteria should be refined by application on uses-cases with different levels of complexity.

V. Conclusion and future work

In this paper, we proposed a preliminary set of trustworthy properties for Knowledge-Based Components. These properties were identified through an analysis of consensus-based trustworthy-AI principles and a risk-based approach, and then specialized for the context of symbolic-AI systems. We acknowledge that these properties may be excessive for simple KBCs, where a requirement-based approach may suffice, but they become necessary for more complex applications.

Our approach is grounded in engineering practices for AI development. To advance this work, we encourage the research community to further refine the definition of trustworthy properties and, in particular, to develop their mathematical formalization and associated verification methods. Such contributions would significantly strengthen the reliability and trustworthiness of knowledge-based systems.

Our future work focuses on implementing verification methods associated with the trustworthy properties identified in this study. We are developing software tools to support automated verification, such as metric-based assessments—as well as methodologies to guide design reviews of Knowledge Models. These methods and tools are being exercised on several industrial use cases exhibiting different levels of complexity.

References

[1] Coll., *Ai act*, https://artificialintelligenceact.eu/, 2022. Accessed: Mar. 5, 2026.

[2] M. Zenner et al., "Tetra–from methodology to operational tools for water-based ai projects", Copernicus Meetings, Tech. Rep., 2026.

[3] G. Schreiber, B. Wielinga, and J. Breuker, *KADS: A principled approach to knowledge-based system development*. Academic Press, 1993, vol. 11.

[4] G. Schreiber, B. Wielinga, W. Jansweijer, et al., "The kactus view on the 'o'word", in *IJCAI workshop on basic ontological issues in knowledge sharing*, vol. 8145, 1995.

[5] M. Uschold et al., "Building ontologies: Towards a unified methodology", *Technical report-university of Edinburgh artificial intelligence applications institute AIAI TR*, 1996.

[6] M. Fernandez, A. Gomez-Perez, and M. Juristo N, "From ontological art towards ontological engineering", in *Proceedings of the Spring Symposium Series on Ontological Engineering (AAAI'97), AAAI Press*, 1997.

[7] W. Ceusters, "Towards a realim-based metric for quality assurance in ontology matching", in *Formal Ontology in Information Systems: Proceedings of the Fourth International Conference (FOIS 2006)*, B. Bennett and C. Fellbaum, Eds., IOS Press, vol. 150, 2006, p. 321.

[8] J. Blázquez, M. Fernández, J. García-Pinar, and A. Gómez-Pérez, "Building ontologies at the knowledge level using the ontology design environment", in *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'98)*, vol. 2, University of Calgary, Alberta, Canadá, Apr. 1998. Accessed: Mar. 5, 2026. [Online]. Available: https://oa.upm.es/6457/.

[9] M. Fernández López and A. Gómez-Pérez, "The integration of ontoclean in webode", in *CEUR Workshop Proceedings*, 2002.

[10] A. Vizedom et al., "Toward ontology evaluation across the lifecycle", *Applied ontology*, vol. 8, pp. 179–194, Oct. 2013. DOI: 10.3233/AO-130125.

[11] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?", *International Journal of Human-Computer Studies*, vol. 43, no. 5, pp. 907–928, 1995, ISSN: 1071-5819. DOI: https://doi.org/10.1006/ijhc.1995.1081. Accessed: Mar. 5, 2026. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1071581985710816.

[12] B. Xue and L. Zou, "Knowledge graph quality management: A comprehensive survey", *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 5, pp. 4969–4988, 2023. DOI: 10.1109/TKDE.2022.3150080.

[13] J. Mattioli, L. Mattioli, and M. Gonzalez, "A brief overview of key quality metrics for knowledge graph solution illustration on digital notams", *Proceedings of the AAAI Symposium Series*, vol. 7, pp. 206–213, Nov. 2025. DOI: 10.1609/aaaiss.v7i1.36888.

[14] J. Pittman, L. Eddy, and K. Wiseman, "Responsible reasoning - a systematic review", *Preprints*, Oct. 2024. DOI: 10.20944/preprints202410.0985.v1. Accessed: Mar. 5, 2026. [Online]. Available: https://doi.org/10.20944/preprints202410.0985.v1.

[15] C. Michel–Delétie and M. K. Sarker, "Neuro-symbolic methods for trustworthy ai: A systematic review with a focus on interpretability", *Neurosymbolic Artificial Intelligence 0*, 2024.

[16] A. Agiollo and A. Omicini, "Measuring trustworthiness in neuro-symbolic integration", in *2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS)*, 2023, pp. 1–10. DOI: 10.15439/2023F6019.

[17] F. Mamalet et al., "White paper machine learning in certified systems", IRT Saint Exupéry ; ANITI, Research Report, Mar. 2021. [Online]. Available: https://hal.science/hal-03176080.

[18] J. Mattioli et al., "An overview of key trustworthiness attributes and kpis for trusted ml-based systems engineering", *AI and Ethics*, vol. 4, no. 1, pp. 15–25, 2024.

[19] Y. Ye, X. Cui, and D. Ouyang, "Extracting a justification for owl ontologies by critical axioms", *Frontiers of Computer Science*, vol. 14, no. 4, p. 144 305, 2020.

[20] C. Project, *Confiance.ai body of knowledge*, https://bok.confiance.ai/. Accessed: Mar. 5, 2026.

[21] R. David, D. Habgood, A. Seaborne, and S. Steyskal, *W3c : Shacl12 rules*, https://www.w3.org/TR/shacl12-rules/. Accessed: Mar. 5, 2026.

[22] B. Lantow, "Ontometrics: Putting metrics into use for ontology evaluation.", in *KEOD*, 2016, pp. 186–191.

[23] C. Laudy and N. Museux, "Peacock: A benchmarks generation framework for high-level information fusion evaluation", in *Fusion 21*, Nov. 2021. DOI: 10.23919/FUSION49465.2021.9627038.