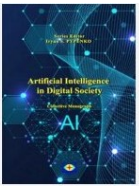
Artificial Intelligence in Digital Society,
Volume 2, 2027
DOI: 10.26697/9786177089208.2027
ISBN 978-617-7089-20-8 (Vol. 2)
ISBN 978-617-7089-18-5 (Series)



Chapter 2.
Knowledge Representation in Artificial Intelligence: Concepts, Evolution, Techniques, and Applications

Mohd Zahid M. S. [1], **Sh-Hussain S. H.** [1]

[1] *Open University Malaysia, Malaysia*

Received: *07.09.2026;* ***Accepted:*** *10.02.2027;* ***Published:*** *10.03.2027*
Published online: *27.09.2026*

Abstract
Symbolic artificial intelligence reasons transparently from explicit rules, but those rules must be authored by hand and cope poorly with unfamiliar cases. Statistical methods learn from data and generalise well, yet leave their knowledge distributed across parameters that cannot be inspected or edited. Hybrid systems combining the two are the common response to the limitations of each, and building one requires deciding which knowledge to hold explicitly and which to leave learned. Knowledge representation is the sub-field concerned with the first of these – encoding facts, concepts, relationships, and rules in a form a machine can store, interrogate, and reason over. This chapter treats those foundations in depth: the kinds of knowledge a system holds, the vocabulary of entities, relations, rules, and inference, and the tension between how much a representation can express and how quickly it can be reasoned over. That tension has shaped where machine knowledge is kept, moving from hand-authored rules to formal logic, then to shared ontologies and knowledge graphs, and lately into model parameters. Six families of technique are examined with worked examples: logic and rules, frames and ontologies, probabilistic representation, knowledge graphs, learned embeddings, and temporal and spatial representation. Deliberate representation is what allows a claim to be checked, updated, and traced to its source, even when a model can produce it from memory.

Keywords: knowledge representation, ontology, knowledge graph, inference, hybrid AI, agentic systems.

Cite this chapter as:
Mohd Zahid, M. S., & Sh-Hussain, S. H. (2027). Knowledge representation in artificial intelligence: Concepts, evolution, techniques, and applications. In I. S. Pypenko (Ed.), *Artificial Intelligence in Digital Society, Vol. 2* (pp. 01–15). KRPOCH. https://doi.org/10.26697/aids.2027.2
The electronic version of this chapter is complete. It can be found online in the AIDS Archive https://doi.org/10.26697/aids

This is an Open Access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (http://creativecommons.org/licenses/by/4.0/deed.en).

Introduction

Artificial intelligence has developed along two broad lines. Symbolic systems hold what they know as statements a person can read, check, and edit. Statistical systems learn from data, with the result distributed across numerical parameters that resist inspection. Neither system has proved sufficient alone, which is why working systems increasingly combine the two (d'Avila Garcez & Lamb, 2023). Such combination raises a question the architecture itself does not settle: which knowledge should a system hold explicitly, and which should it learn?

Knowledge representation (KR) is the field concerned with how Symbolic systems store and manipulate information. It is the theoretical and practical backbone of Symbolic AI. However, KR as a field has evolved to also address how knowledge can be structured and integrated into modern statistical and hybrid AI systems. It studies how facts, concepts, relationships, and rules can be encoded so that a machine can store them, search them, and draw conclusions from them (Davis et al., 1993; Russell & Norvig, 2021). In modern AI, KR also encompasses distributed representations such as word embeddings (Carlson et al., 2025; Pennington et al., 2014) or latent vectors in neural networks.

The consequences of that encoding become clear through an example used throughout this chapter. A university records that a student must pass Discrete Mathematics before enrolling in Algorithms. A spreadsheet holds this as a pair of cells. A logical system states it as a rule, which can be chained with other rules to derive a complete study path. A neural model absorbs it into weights learned from thousands of enrolment histories. All three hold the same fact. Only the second can explain why a particular student was blocked from enrolling, and only the first two can be corrected by editing a single entry.

Representation therefore determines three things: what a system can infer beyond what it was told, what it can explain about its own conclusions, and what it can be held accountable for when those conclusions affect people.

This chapter covers the foundations of knowledge representation, the shift in where machine knowledge has been kept, six families of technique illustrated on a common example, and the reasons explicit representation remains necessary in systems that also learn.

Foundations of Knowledge Representation

Data, Information, and Knowledge

The three terms are often used interchangeably, though they mark distinct levels. Data are uninterpreted symbols: the string CBMA2103 and the number 65. Information is data placed in a context that gives it meaning: student 12345 scored 65 in CBMA2103. Knowledge adds the general statements that let new conclusions be drawn from information: a score of 50 or above is a pass, and a pass in CBMA2103 satisfies the prerequisite for CBSP2103.

The distinction matters because only the third level supports inference. A database can report the score. A knowledge base can determine, without being told, that the student may now enrol.

Kinds of Knowledge

Declarative knowledge states what is true. The prerequisite relation above is declarative: it asserts a fact about the world and says nothing about how to use it. Procedural knowledge states how to act. The sequence of checks an academic advisor performs when confirming a student's eligibility is procedural.

Heuristic knowledge captures rules of thumb that work well without being guaranteed. An advisor might know that students who fail a mathematics course twice rarely succeed on a third attempt. Expert systems of the 1970s and 1980s were built largely from knowledge of this kind (Buchanan & Shortliffe, 1984).

Meta-knowledge is what a system knows about its own knowledge: which facts are reliable, which are out of date, and where its coverage ends. Systems lacking it cannot distinguish a confident answer from a guess, a limitation that carries directly into contemporary language models.

The Vocabulary of Representation

Most representations are built from a small set of primitives. Entities are the things a system reasons about: students, courses, semesters. Attributes are properties of entities, such as a course code or credit value. Relations connect entities, as in Algorithms has-prerequisite Discrete Mathematics. Rules state general conditions under which conclusions follow. Inference is the process of deriving statements that were never explicitly stored.

Two properties of an inference procedure deserve attention. A procedure is sound if everything it derives is true given what it was told. It is complete if it derives everything that is true given what it was told. Soundness protects against false conclusions, while completeness ensures none are missed. Rich representations frequently force a choice between the two, or make complete reasoning too slow to run.

The difference between a stored fact and a derived one is central. Suppose the prerequisite chain runs from Mathematics for IT to Discrete Mathematics to Algorithms. A system holding only the adjacent pairs can still derive that the first course is required, indirectly, for the third. That conclusion was never stored. Representation is what makes such derivation possible.

The Several Roles a Representation Plays

Davis, Shrobe, and Szolovits (1993) observed that any representation performs five roles at once, each imposing its own demands.

A representation is a surrogate: a stand-in for things in the world, and necessarily an imperfect one. It is a set of ontological commitments, a decision about which distinctions matter, taken before any reasoning occurs. Choosing to

model courses but not individual class sessions determines what can subsequently be asked. It is a theory of intelligent reasoning, an implicit claim about which conclusions should follow from what. It is a medium for efficient computation, since a representation nobody can compute with has no practical use. It is a medium of human expression, readable by the people who build, audit, and maintain it.

These roles conflict. A representation rich enough to capture fine ontological distinctions is often too slow to compute with. One optimised for computation may become unreadable. Design in this field consists largely of managing that conflict.

Questions That Recur Across Formalisms

Four questions arise regardless of which technique is chosen.

The first concerns expressiveness against tractability. Richer representations permit more to be said, at the cost of slower reasoning. First-order logic can express almost any relationship, though determining what follows from a set of first-order statements is undecidable in general. Description logics deliberately restrict expressiveness to guarantee that reasoning terminates (Baader et al., 2017). Neither choice is correct in the abstract.

The second concerns absent information. Under the closed-world assumption, anything not stated is taken to be false, so a student with no recorded pass is treated as not having passed. Under the open-world assumption, the same absence means only that the system does not know. Database systems generally adopt the first, ontology languages the second. Applying the wrong assumption produces conclusions that are confident and incorrect.

The third concerns revision. Classical logic is monotonic, meaning that adding facts never withdraws earlier conclusions. Practical reasoning is not. A student provisionally eligible for a course ceases to be eligible once a failed resit is recorded. Non-monotonic formalisms allow conclusions to be retracted as knowledge grows.

The fourth concerns meaning. Symbols inside a machine acquire significance only through their connection to the world outside it. A system manipulating the token Algorithms has no access to what the course involves. This is the symbol grounding problem (Harnad, 1990), and it becomes acute when representations are learned from data rather than authored by hand.

Where Machine Knowledge Has Been Kept

Rather than recount the history of artificial intelligence, which is treated elsewhere in this volume, this section follows one thread: the changing answer to where a system's knowledge resides.

Early symbolic systems kept it in hand-authored structures. Semantic networks represented concepts as nodes and relations as labelled links (Quillian,

1968). Frames grouped what was known about a concept into slots carrying default values (Minsky, 1974). Production rules encoded expertise as condition-action pairs, and expert systems such as MYCIN showed this was sufficient for narrow diagnostic tasks (Buchanan & Shortliffe, 1984). Knowledge lived in files a person had written and could revise.

Formal logic relocated knowledge into declarative statements with defined semantics. Description logics later constrained expressiveness in exchange for decidable reasoning, supplying the theoretical basis for ontology languages (Baader et al., 2017).

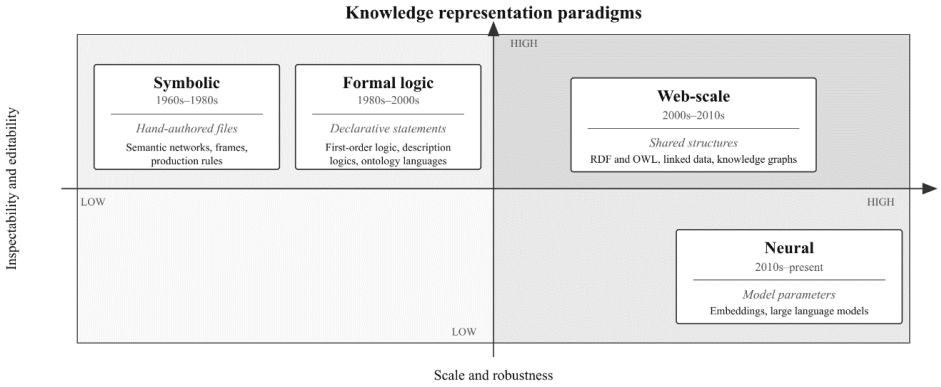
Web-scale work moved knowledge out of individual systems into shared structures. The Resource Description Framework (RDF) represented facts as subject-predicate-object triples, and the Web Ontology Language (OWL) added the means to define classes and constraints over them (Berners-Lee et al., 2001; W3C OWL Working Group, 2012). Knowledge graphs built on these foundations now underpin search, commerce, and biomedical research (Hogan et al., 2021).

Neural methods dissolved knowledge into parameters. Embeddings place entities and relations in a continuous vector space, where similarity and even relational structure can be computed arithmetically (Bordes et al., 2013; Ji et al., 2022). Large language models extend this further, holding substantial knowledge implicitly in weights that nobody authored and nobody can directly edit.

The trajectory runs from knowledge that is fully explicit and fully editable toward knowledge that is neither. Each step gained scale and robustness. Each also surrendered a degree of inspectability. Figure 2.1 summarises this shift.

Figure 2.1

Where Machine Knowledge Has Been Kept



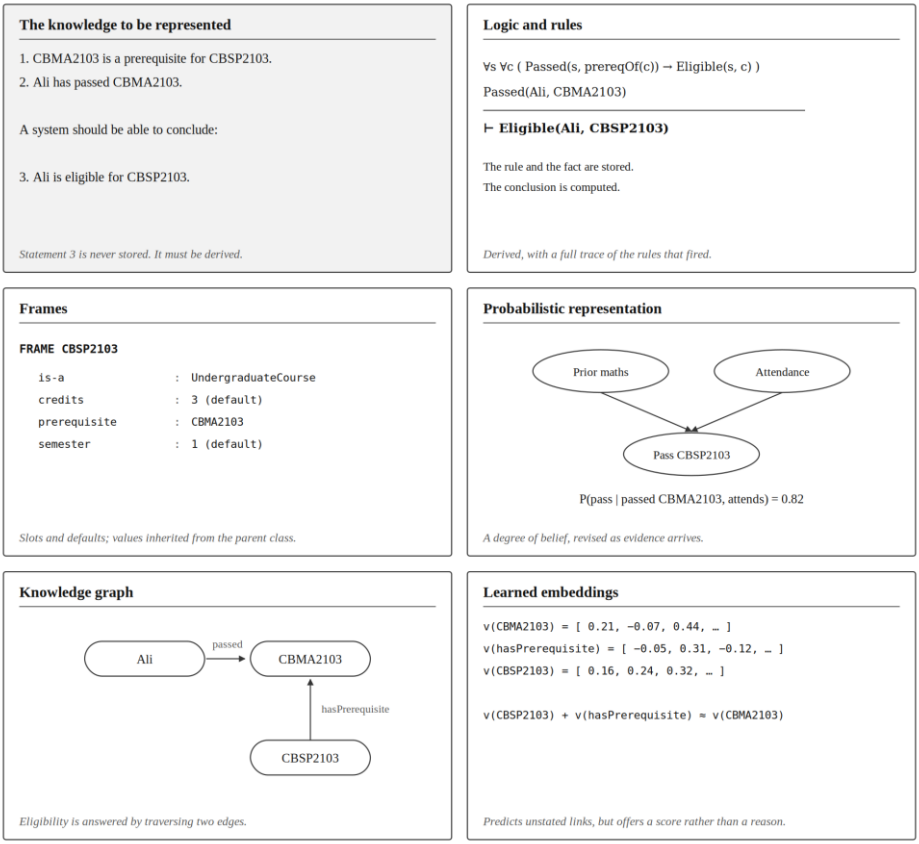
Note. Each era relocated knowledge to a new medium. Scale and robustness rose across the sequence; the ease with which a person could inspect or edit what the system knew fell.

Techniques for Representing Knowledge

Each technique below is applied to the same small domain: a university curriculum with courses, prerequisites, students, and enrolment decisions. Holding the domain constant makes the differences between formalisms visible. Figure 2.2 places five of the encodings side by side; temporal and spatial representation is omitted because the example involves no change over time. The subsections that follow examine each technique in turn.

Figure 2.2

One Body of Knowledge Encoded Five Ways



Note. The shaded panel states the knowledge to be represented. Each remaining panel encodes the same two facts in a different formalism, with the derived conclusion each supports. The formalisms differ less in what they can store than in what they can conclude and justify.

Logic and Rules

First-order logic represents knowledge as statements over entities, predicates, and quantifiers. The prerequisite rule becomes:

$\forall s\ \forall c$ *(Passed(s, prerequisiteOf(c)) → Eligible(s, c))*

Production rules express the same knowledge in condition-action form, which domain experts generally find easier to author:

IF student has passed CBMA2103 THEN student is eligible for CBSP2103

Given the stored fact Passed(Ali, CBMA2103), both formalisms derive Eligible(Ali, CBSP2103). Neither was told this directly.

A rule-based system has three components: a rule base holding general knowledge, a working memory holding facts about the case at hand, and an inference engine that applies the former to the latter. Forward and backward chaining are two control strategies available to that engine. They share the same rule base and differ in what is placed in working memory, and when.

Two rules serve for illustration:

R1: IF Score(s, c) ≥ 50 THEN Passed(s, c)

R2: IF Passed(s, c) AND prerequisiteOf(d) = c THEN Eligible(s, d)

Forward chaining takes a set of facts as its input, asserted into working memory before the engine runs. No question is posed. Asserting

F1: Score(Ali, CBMA2103) = 65

F2: prerequisiteOf(CBSP2103) = CBMA2103

sets the engine going. It cycles through the rule base, firing every rule whose conditions the current contents of working memory satisfy, and adding the results:

Cycle 1: R1 matches F1, since 65 ≥ 50. The engine adds Passed(Ali, CBMA2103).

Cycle 2: R2 now matches, binding c to CBMA2103 and d to CBSP2103. The engine adds Eligible(Ali, CBSP2103).

Cycle 3: no rule matches. The engine halts.

Working memory now holds four facts where it began with two. The eligibility conclusion was derived without anyone asking for it, alongside whatever else the rules supported.

Backward chaining takes a goal as its input: a statement whose truth is to be established. Facts are consulted only as the proof requires them. Asked whether

G: Eligible(Ali, CBSP2103)

holds, the engine searches for a rule whose conclusion matches the goal:

Step 1: R2 concludes Eligible, so proving G requires two subgoals, prerequisiteOf(CBSP2103) = c and Passed(Ali, c).

Step 2: F2 is already stored and satisfies the first, binding c to CBMA2103.

Step 3: Passed(Ali, CBMA2103) is not stored, so it becomes a subgoal in turn. R1 concludes Passed, leaving Score(Ali, CBMA2103) ≥ 50 to establish.

Step 4: F1 satisfies it. Every subgoal is established, so G holds.

A subgoal matching no rule and no stored fact need not end the proof. Predicates marked as askable are put to the user as questions, which is how

diagnostic expert systems conducted consultations. The goal determines which questions are worth asking, so the user is never asked about matters irrelevant to it.

The two input styles suit different situations. Forward chaining fits cases where facts arrive over time and any consequence may matter, such as monitoring enrolment records to flag students who have become eligible. Backward chaining fits cases where a specific question is put and only the facts bearing on it need be gathered, such as an advisor checking whether one named student may enrol.

Frames and Ontologies

Frames organise knowledge around concepts rather than statements. A Course frame holds slots for code, credits, prerequisites, and level, with defaults where values are typically shared. Inheritance allows a Postgraduate Course frame to inherit from Course while overriding the credit default.

Ontologies formalise this arrangement. An ontology specifies the classes in a domain, the relations between them, and constraints on those relations, expressed in a language with defined semantics (Gruber, 1993; Studer et al., 1998). In OWL, the curriculum might state that Course and Student are disjoint classes, that hasPrerequisite is transitive, and that every Postgraduate Course is a Course.

The choice of ontology profile affects computational cost (Motik et al., 2012). The lightweight OWL 2 profiles (EL, QL, and RL) trade expressiveness for tractable reasoning. OWL 2 DL guarantees decidable reasoning by restricting the logic to description logic. OWL 2 Full allows maximum expressiveness but sacrifices guaranteed termination.

Transitivity earns its place here. Declaring hasPrerequisite transitive means a reasoner derives that CDMA2103 is a prerequisite of CBSP2103 from two separately stated links, with no rule written for the chain.

Probabilistic Representation

Bayesian networks represent knowledge as a directed graph of variables with associated conditional probability distributions (Pearl, 1988). For the curriculum domain, prior mathematics attainment, hours studied, and attendance might jointly influence the probability of passing Algorithms. Given evidence that a student attended irregularly, the network updates the probability of failure and propagates that change to downstream variables such as expected time to graduation.

Markov Logic Networks combine first-order logic with Markov networks (Richardson & Domingos, 2006). They attach weights to logical formulas. A formula that is always true in the domain receives a high weight. A formula that is frequently violated receives a low weight. This allows the system to handle exceptions to strict logical rules.

Knowledge Graphs

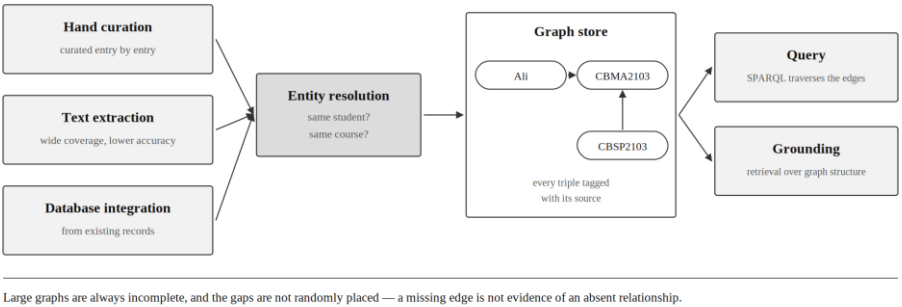
A knowledge graph stores facts as triples. Part of the curriculum becomes:

(CBSP2103, hasPrerequisite, CBMA2103)
(Ali, enrolledIn, CBMA2103)
(CBMA2103, offeredIn, Semester1)

Queries traverse the graph. SPARQL, the W3C query language for RDF, matches triple patterns against the stored graph and returns the bindings that satisfy them (Harris & Seaborne, 2013). A SPARQL query can retrieve every course for which a given student is eligible by following prerequisite edges and checking pass records. Figure 2.3 illustrates how a knowledge graph is built and used.

Figure 2.3

How a Knowledge Graph Is Built and Used



Note. Sources are reconciled by entity resolution before entering the graph, and every triple retains its source so that claims of different reliability can be told apart.

Learned Embeddings

Embedding methods map entities and relations to vectors learned from existing data. TransE, for example, trains vectors such that the embedding of a subject plus the embedding of a relation approximates the embedding of the object (Bordes et al., 2013). RotatE maps relations as rotations in complex vector space, which handles symmetric, antisymmetric, inverse, and compositional relations better than translation-based models (Sun et al., 2019).

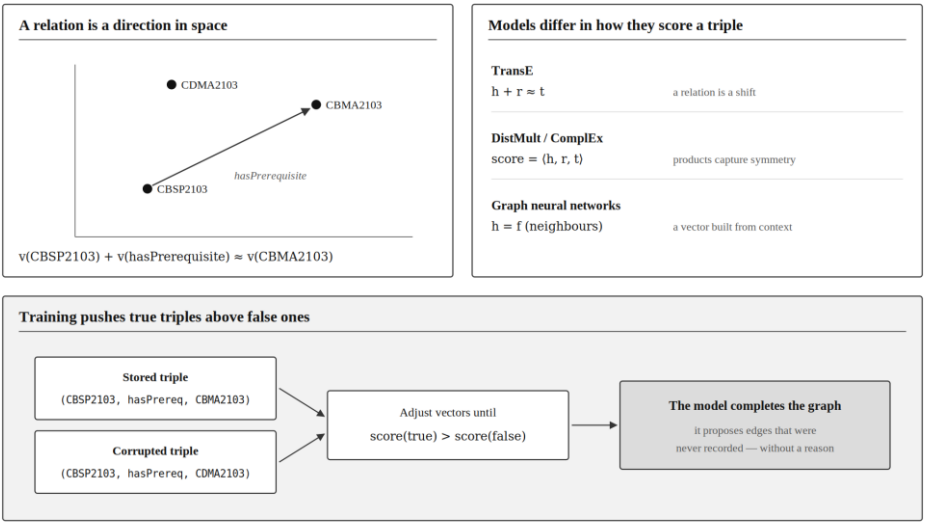
In TransE, h, r and t denote the vectors of the head (subject), the relation and the tail (object) of a triple. The model judges a triple by how far h + r lands from t. Its score is the negative of this distance, $-\|h + r - t\|$, so a triple that fits the pattern scores close to zero, while one that does not fit scores strongly negative. Other models use different scoring functions. DistMult (Yang et al., 2015) and ComplEx (Trouillon et al., 2016) multiply the three vectors element by element and sum the result, whereas graph neural networks first build each entity's vector from its neighbours before scoring (Figure 2.4).

Training begins with random vectors and relies on two kinds of example. A stored triple is a fact already recorded in the graph, such as (CBSP2103, hasPrerequisite, CBMA2103), and is treated as true. A corrupted triple is produced by replacing the head or tail of a stored triple with a randomly chosen entity, such

as (CBSP2103, hasPrerequisite, CDMA2103). The graph does not record this fact, so the model treats it as false, even though a few corrupted triples may later turn out to be true.

Figure 2.4

How Embedding Models Represent and Complete a Graph



Note. Entities and relations become vectors, models differ in how they score a triple, and training separates true triples from corrupted ones. The result proposes edges that were never recorded.

For each pair, the model computes both scores. It incurs a loss whenever the corrupted triple scores higher than the stored one or falls within a chosen margin of it. Gradient descent then nudges the vectors involved to reduce this loss, moving h + r closer to the true tail and further from the false one. Repeated over many thousands of pairs, these small adjustments arrange the vector space so that score(true) > score(false) for most recorded facts. A triple absent from the graph that still scores highly becomes a candidate edge, which is how the model proposes links that were never recorded.

Trained on enrolment records, such a model can predict that a course not yet linked to any prerequisite probably requires a mathematics background, on the basis of its similarity to courses that carry one.

The same principle underlies the knowledge held inside large language models, at far greater scale and with no explicit graph at all. Both trade the ability to state and audit a single fact for the ability to generalise beyond what was recorded.

Temporal and Spatial Representation

Many real-world domains require knowledge to be anchored in time and space. Allen's interval algebra defines thirteen basic relations between time intervals, such as before, meets, overlaps, and during (Allen, 1983). A temporal knowledge graph adds a time dimension to standard triples.

(Ali, enrolledIn, CBMA2103, 2024)

This structure allows the system to reason about state changes. A student might be eligible for a course in one semester and ineligible in the next due to a failed prerequisite. Standard static graphs cannot capture this shift without duplicating entities.

Comparing the Techniques

Table 2.1 rates the techniques against the design criteria. Table 2.2 states what each can and cannot establish. No column dominates, so the practical question is which properties a given application cannot do without.

Table 2.1

Comparison of Knowledge Representation Techniques

Technique	Expressiveness	Reasoning cost	Uncertainty	Explainability	Tolerates gaps
Logic and rules	High	High for full first-order logic	No	High; full inference trace	Low
Frames and ontologies	Moderate; deliberately decidable	Moderate	No	High; subsumption is traceable	Moderate
Bayesian networks	Moderate	Grows sharply with variables	Yes	Moderate	Moderate
Knowledge graphs	Low without an ontology	Low for traversal	No	High; triples carry provenance	Low
Learned embeddings	Implicit	Low at query time	Partial; scores only	Low	High
Temporal and spatial	Moderate	Rises with time granularity	No	High; facts are time-stamped	Moderate

Note. No technique dominates across all criteria. Hybrid systems combine techniques so that the properties an application requires are each supplied by some component.

Selection follows from the requirement. Applications that must explain decisions to affected parties need explicit rules or ontologies. Applications operating under genuine uncertainty need probabilistic representation. Applications integrating heterogeneous sources need graphs. Applications that must generalise beyond recorded facts need learned representations. Systems facing several of these requirements at once combine techniques, which is the practical content of hybrid design.

Table 2.2

What Each Technique Can and Cannot Conclude

Technique	What it can conclude	What it cannot
Logic and rules	Anything entailed by the stated rules, with a trace of which rules fired	Degree or likelihood; a rule either fires or does not
Frames and ontologies	Class membership, subsumption, and inconsistency	Uncertainty; open-world gaps yield weaker conclusions than expected
Probabilistic representation	Calibrated degrees of belief, revised as evidence arrives	Scale without costly data or expert probability estimates
Knowledge graphs	Anything reachable by traversal, across data merged from several sources	Enforce meaning unaided; a missing edge looks like an absent relation
Learned embeddings	Plausible facts never stated, including completions of an incomplete graph	Justify a prediction; correcting an error means retraining
Temporal and spatial	The sequence of events and the validity of facts at specific moments	Continuous time without discretisation, which introduces boundary errors

Note. The entries restate, in compact form, the reasoning limits. Table 2.1 rates the techniques; this table states what they can establish.

Representation, Accountability, and Digital Society

In a digital society, many decisions that once passed through a person now pass through a system: whether a student may enrol, whether a loan is approved, which applicants reach an interview. The representation inside such a system determines whether an affected person can ask why a decision was made, and whether anyone can answer. Contemporary architectures differ in how well they preserve that ability.

Retrieval-augmented generation is one response. It supplies a model with retrieved documents at inference time rather than relying on parametric memory, partly because provenance and updateability remain unsolved for knowledge held in parameters (Lewis et al., 2020). Graph-based variants retrieve over structured knowledge rather than flat text (Edge et al., 2024). Machine learning also constructs representations, extracting entities and relations from text to populate graphs at a scale hand-authoring cannot reach.

Autonomous and agentic systems depend on representations that persist. An agent executing a multi-step task needs a model of its environment, schemas describing the actions available to it, and memory surviving beyond a single interaction. Much of this is knowledge representation under a different name, and systems leaving it implicit prove correspondingly difficult to debug or constrain.

The reason this matters beyond engineering is that explicit representation is what makes a claim checkable. A conclusion derived from stated rules can be traced to those rules. A fact in a knowledge graph carries a source and can be corrected by editing one triple. A conclusion produced from model parameters offers neither trace nor edit point. When automated decisions affect enrolment, credit, employment, or care, the difference between these two situations determines

whether a decision can be contested, audited, or corrected at all. None of this favours symbolic methods over learned ones. It identifies instead what must be preserved when the two are combined: enough explicit structure that consequential claims remain traceable to something a person can inspect.

Conclusion

Knowledge representation determines what an artificial intelligence system can infer, explain, and be held accountable for. Its foundations are well established: the distinction between data and knowledge, the vocabulary of entities and relations, the several roles any representation plays, and the recurring tension between expressiveness and tractability. Highly expressive representation languages allow for detailed modelling of complex domains, but they often make automated reasoning computationally intractable. Simpler languages are computationally efficient but lack the capacity to represent complex relationships. Its techniques offer different trade-offs rather than a ranking, as Table 2.1 shows.

The shift toward knowledge held implicitly in model parameters has brought scale and robustness at the cost of inspectability. Readers designing systems for consequential domains should treat the division between explicit and learned knowledge as a deliberate decision rather than a by-product of architecture. Open problems remain in grounding generated claims, reasoning over incomplete graphs, and representing causal rather than merely correlational structure.

Ethical Approval

This chapter is a conceptual and review contribution. It did not involve human participants or animals, and ethical approval was therefore not required.

Funding Source

This research did not receive any outside funding or support.

References

Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM, 26*(11), 832–843. https://doi.org/10.1145/182.358434

Baader, F., Horrocks, I., Lutz, C., & Sattler, U. (2017). *An introduction to description logic*. Cambridge University Press. https://doi.org/10.1017/9781139025355

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American, 284*(5), 34–43. https://doi.org/10.1038/scientificamerican0501-34

Bordes, A., Usunier, N., García-Durán, A., Weston, J., & Yakhnenko, O. (2013). Translating embeddings for modeling multi-relational data. *Advances in Neural Information Processing Systems, 26*, 2787–2795. https://dl.acm.org/doi/10.5555/2999792.2999923

Buchanan, B. G., & Shortliffe, E. H. (1984). *Rule-based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project*.

Addison-Wesley. https://people.dbmi.columbia.edu/~ehs7001/Buchanan-Shortliffe-1984/MYCIN%20Book.htm

Carlson, R., Bauer, J., & Manning, C. D. (2025). *A new pair of GloVes*. arXiv. https://doi.org/10.48550/arXiv.2507.18103

D'Avila Garcez, A., & Lamb, L. C. (2023). Neurosymbolic AI: The 3rd wave. *Artificial Intelligence Review, 56*(11), 12387–12406. https://doi.org/10.1007/s10462-023-10448-w

Davis, R., Shrobe, H., & Szolovits, P. (1993). What is a knowledge representation? *AI Magazine, 14*(1), 17–33. https://doi.org/10.1609/aimag.v14i1.1029

Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., & Larson, J. (2024). *From local to global: A graph RAG approach to query-focused summarization*. arXiv. https://doi.org/10.48550/arXiv.2404.16130

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition, 5*(2), 199–220. https://doi.org/10.1006/knac.1993.1008

Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena, 42*(1–3), 335–346. https://doi.org/10.1016/0167-2789(90)90087-6

Harris, S., & Seaborne, A. (Eds.). (2013). *SPARQL 1.1 query language (W3C Recommendation).* World Wide Web Consortium. https://www.w3.org/TR/sparql11-query/

Hogan, A., Blomqvist, E., Cochez, M., d'Amato, C., de Melo, G., Gutiérrez, C., Kirrane, S., Labra Gayo, J. E., Navigli, R., Neumaier, S., Ngonga Ngomo, A.-C., Polleres, A., Rashid, S. M., Rula, A., Schmelzeisen, L., Sequeda, J., Staab, S., & Zimmermann, A. (2021). Knowledge graphs. *ACM Computing Surveys, 54*(4), Article 71. https://doi.org/10.1145/3447772

Ji, S., Pan, S., Cambria, E., Marttinen, P., & Yu, P. S. (2022). A survey on knowledge graphs: Representation, acquisition, and applications. *IEEE Transactions on Neural Networks and Learning Systems, 33*(2), 494–514. https://doi.org/10.1109/TNNLS.2021.3070843

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive NLP tasks. *Advances in Neural Information Processing Systems, 33*, 9459–9474. https://doi.org/10.48550/arXiv.2005.11401

Minsky, M. (1974). *A framework for representing knowledge* (AI Memo No. 306). MIT Artificial Intelligence Laboratory. https://courses.media.mit.edu/2004spring/mas966/Minsky%201974%20Framework%20for%20knowledge.pdf

Motik, B., Cuenca Grau, B., Horrocks, I., Wu, Z., Fokoue, A., & Lutz, C. (Eds.). (2012). *OWL 2 Web Ontology Language: Profiles (W3C Recommendation).* World Wide Web Consortium. https://www.w3.org/TR/owl2-profiles/

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. Morgan Kaufmann. https://doi.org/10.1016/B978-0-08-051489-5.50008-4

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1532–1543). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

Quillian, M. R. (1968). Semantic memory. In M. Minsky (Ed.), *Semantic Information Processing* (pp. 227–270). https://philpapers.org/rec/MINSIP

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Machine Learning, 62*(1–2), 107–136. https://doi.org/10.1007/s10994-006-5833-1

Russell, S. J., & Norvig, P. (2021). *Artificial intelligence: A modern approach* (4th ed.). Pearson. https://www.pearson.com/en-us/subject-catalog/p/artificial-intelligence-a-modern-approach/P200000003500?view=educator

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data & Knowledge Engineering, 25*(1–2), 161–197. https://doi.org/10.1016/S0169-023X(97)00056-6

Sun, Z., Deng, Z.-H., Nie, J.-Y., & Tang, J. (2019). *RotatE: Knowledge graph embedding by relational rotation in complex space*. Paper presented at the 7th International Conference on Learning Representations (ICLR 2019), New Orleans, LA. https://openreview.net/forum?id=HkgEQnRqYQ

Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., & Bouchard, G. (2016). Complex embeddings for simple link prediction. In *Proceedings of the 33rd International Conference on Machine Learning* (Proceedings of Machine Learning Research, Vol. 48, pp. 2071–2080). PMLR. https://proceedings.mlr.press/v48/trouillon16.html

W3C OWL Working Group. (2012). *OWL 2 Web Ontology Language document overview* (2nd ed.). World Wide Web Consortium. https://www.w3.org/TR/owl2-overview/

Yang, B., Yih, W., He, X., Gao, J., & Deng, L. (2015). *Embedding entities and relations for learning and inference in knowledge bases*. Paper presented at the 3rd International Conference on Learning Representations (ICLR 2015), San Diego, CA. https://arxiv.org/abs/1412.6575

Information about the authors:

Mohd Zahid Mohd Soperi – https://orcid.org/0000-0001-5042-5793; PhD, Associate Professor, Faculty of Computing and Analytics, Open University Malaysia, Petaling Jaya, Selangor, Malaysia.

Sh-Hussain Siti Hadrina – https://orcid.org/0009-0006-7910-8859; PhD, Lecturer, Faculty of Computing and Analytics, Open University Malaysia, Petaling Jaya, Selangor, Malaysia.