

Verifiable reasoning of large language models over structured spatial data: a survey and research agenda

Kamal Essam and Ryszard S. Romaniuk

Abstract—Large language models increasingly answer questions grounded in structured spatial data such as road networks: in navigation assistants, in conversational interfaces to geographic information, and in tools that maintain digital maps. Their answers are fluent but not verifiable, and in spatial domains a claim about a turn restriction, a lane, or an opening time is either consistent with the map or it is not. This survey reviews the methods by which such claims can be machine-checked. We organize the literature along an argument: models cannot reliably verify their own outputs; external checkers such as program execution and symbolic solvers demonstrably can; constrained decoding guarantees syntactic but not factual validity; execution against the data is the established gold standard of evaluation in the text-to-query literature; and knowledge-graph grounding already supports detect-verify-repair pipelines. We then examine the spatial domain specifically and find a documented gap: existing geospatial benchmarks measure model capability, and none verifies model claims against lane-level, temporal, or version-aware map data. We synthesize a taxonomy of verification mechanisms, compare existing benchmarks by oracle type, and outline a research agenda toward validity contracts and certificates for spatial claims that remain checkable as the underlying map evolves.

Keywords—large language models; verification; road networks; knowledge graphs; benchmarks; spatial data

I. INTRODUCTION

LARGE language models have begun to answer questions whose ground truth lives in structured spatial data. They power conversational interfaces to maps, assist drivers in vehicles, and increasingly operate inside the toolchains that build and maintain digital maps themselves. Their answers are fluent, confident, and unverifiable. The gap between these properties is wider than intuition suggests: on the most comprehensive map-grounded benchmark to date, no foundation model exceeds 67% accuracy, more than 20 points below human performance [1]. In most applications of language models, an imperfect answer is a degraded answer. In spatial domains it is something sharper. A claim about a turn restriction, a lane connection, or an opening time is either consistent with the map or it is not; there is no partially permitted left turn.

K. Essam is with TomTom, Warsaw, Poland (e-mail: kessam1@outlook.com; ORCID: 0009-0009-8854-1613).

R. S. Romaniuk is with the Institute of Electronic Systems, Faculty of Electronics and Information Technology, Warsaw University of Technology, Warsaw, Poland (e-mail: ryszard.romaniuk@pw.edu.pl; ORCID: 0000-0002-5710-4041).

The same property that makes errors unforgiving also makes the domain unusually well suited to a remedy. Unlike open-domain question answering, spatial reasoning has an authoritative formal world model already in production: the road network as an attributed graph, with topology, restrictions, temporally conditioned attributes, and, in industrial settings, a versioned release history. Reliability therefore need not wait for better models. A claim about the road network can, in principle, be checked: translated into a formal assertion and evaluated against the graph itself.

This survey reviews the methods by which such checking can be done. Geospatial capability of language models has recently been surveyed systematically [2]; that line of work maps what models know about geographic space. The complementary question of how the claims of a language model can be verified against structured spatial data has not, to our knowledge, been systematically reviewed, despite mature verification literatures in neighboring domains. Our aim is to organize those literatures into a usable foundation for spatial verification and to state precisely what is missing. Throughout, we call the external authority that produces a verdict on a claim an *oracle*, and distinguish three oracle types: reference-answer oracles (static gold labels), execution oracles (the verdict is the outcome of running the generated artifact against the data), and graph-evidence oracles (the verdict is entailment by a retrieved subgraph).

The survey makes four contributions. First, it organizes the relevant work as a single argument across five method families: the limits of model self-correction (Section II), external verification architectures (Section III), constrained generation (Section IV), execution-based evaluation (Section V), and grounding in knowledge graphs (Section VI). Second, it compares the benchmarks closest to the problem by oracle type and spatial coverage (Table I), documenting that none covers lane-level topology, turn restrictions, temporal validity, or map-version provenance (Section VII). Third, it synthesizes a taxonomy of verification mechanisms by what they guarantee, when they apply, and what evidence they produce (Table II). Fourth, it derives a research agenda: validity contracts for spatial claims, execution-verified benchmarks regenerated from versioned map releases, and certificates whose verdicts are defined across map versions (Section VIII).

A note on scope and method. This is a narrative, argument-led review, not a systematic one. The corpus was assembled in mid-2026 by keyword search over the ACL Anthology, IEEE



Xplore, Scopus, dblp, and the OpenReview proceedings of the major machine-learning venues, combining verification terms (verification, grounding, execution, constrained decoding, fact checking) with spatial terms (road network, map, geospatial, OpenStreetMap), followed by citation chasing from the papers found. Twenty core papers were selected under three criteria: original research, publication at an indexed venue, and a distinct contribution to the argument of one section. Surveys and preprints are cited as context only, and the limitations of surveyed methods are reported alongside their results. Three neighboring literatures are deliberately out of scope: perception benchmarks from autonomous driving, which verify sensor outputs rather than language-model claims; synthetic spatial-reasoning test sets, which lack an authoritative external data source; and the formalization of traffic rules for monitoring vehicle behavior, which checks trajectories rather than claims. The limits of this selection are discussed at the end of Section VIII.

II. THE LIMITS OF SELF-CORRECTION

An attractive hope drove much of the early work on reliable language models: if a model produces a wrong answer, perhaps the same model can be prompted to find and fix its own mistake. Huang et al. examined this hope under controlled conditions and gave it both a name and a negative result [3]. They define intrinsic self-correction as a setting in which the model critiques and revises its own output without any external feedback, and they show that under this definition self-correction does not improve reasoning accuracy and frequently degrades it. With GPT-3.5, accuracy on CommonsenseQA falls from 75.8% to 38.1% after a single round of self-critique; with GPT-4, accuracy on GSM8K falls from 95.5% to 89.0% over two rounds. Earlier positive reports, they demonstrate, had quietly relied on oracle information: the correction loop stopped when the answer matched the ground-truth label, so the verification problem was being solved by the benchmark rather than by the model. Their answer-change analysis makes the failure concrete. When a model does revise its answer, correct-to-incorrect flips outnumber the reverse.

For reasoning over graph-structured data, the most instructive result is that of Stechly et al. [4], who studied generate-verify-revise loops on constraint-satisfaction and planning problems, including graph coloring, a task structurally close to reasoning over road networks. Self-critique collapsed graph-coloring accuracy from 16% to 2%, while the same loop driven by a sound external verifier raised it to 37%; on Blocksworld planning, the sound verifier more than doubled accuracy, to 87%. Their diagnosis is precise: the model is an unsound verifier of its own solutions, rejecting almost all correct graph colorings (a 95.8% false-negative rate). Notably, on the constraint-satisfaction tasks most of the benefit was retained even when feedback was reduced to a binary accept/reject signal; on Blocksworld, richer error feedback did matter (60% with a binary signal against 87%). A sound judge carries the improvement, more than an eloquent critique.

These negative results should be read fairly. Self-refinement does help where feedback concerns style or preference rather

than fact [5], and a critical survey of the area concludes that the decisive variable is the source of the feedback signal [6]. Verification signals can also be learned: process-supervised reward models judge each step of a reasoning chain and substantially improve mathematical problem solving [7]. A learned verifier, however, is still a model, with error modes of its own. It raises the quality of the judgment but does not ground it in anything outside the model family.

The conclusion we draw is structural. If model claims are to be trusted, the checking signal must come from outside the model, and its authority must derive from something the model cannot influence. The remainder of the survey is therefore organized by where that external signal comes from: a program interpreter or symbolic solver (Section III), a grammar or type system applied during generation (Section IV), execution against the data itself (Section V), and a knowledge graph (Section VI). Structured spatial data, we will argue, is the domain where the strongest form of this signal, execution against an authoritative model of the world, is most naturally available (Sections VII and VIII).

III. EXTERNAL VERIFICATION ARCHITECTURES

If the checking signal must come from outside the model, the first design question is where to put the checker. The literature has converged on three architectures, distinguished by what the model is asked to produce and what external machinery judges it.

The first architecture makes the reasoning itself executable. In program-aided language models, the model does not produce an answer but a program whose execution by a deterministic interpreter yields the answer [8]. On the GSM8K arithmetic benchmark this raised top-1 accuracy to 72.0%, compared with 65.6% for chain-of-thought prompting with the same underlying model. On GSM-Hard, a variant in which one operand is replaced by a large random number, the difference grows to 61.2% against 23.1%, because large numbers break free-form arithmetic but do not trouble an interpreter. The most important evidence, however, is an ablation: when the same model is asked to simulate the interpreter instead of running it, accuracy collapses from 72.0% to 23.2%. The gain comes from the deterministic executor, not from the discipline of writing code. Authority resides in the external engine.

The second architecture translates the problem into a formal representation and delegates judgment to a symbolic solver. Logic-LM formalizes a natural-language reasoning problem, hands it to a deductive or constraint solver, and, critically, uses the solver's error messages to drive a repair loop that revises the formalization [9]. With GPT-4, accuracy on ProofWriter rises from 68.1% under chain-of-thought to 79.7%, with an average relative improvement of 39.2% over standard prompting across five datasets, and solver-guided refinement raises the rate of executable formalizations by about five points. This is the cleanest published instance of generate, verify, and repair against a symbolic authority. Its caveat is equally instructive: on ProntoQA, a dataset easy enough for direct reasoning, chain-of-thought outperforms the formalization route. Translation into logic has a cost, and it pays off only where the checking is genuinely hard.

The third architecture keeps free-form generation but grounds the critique. CRITIC lets the model revise its answers in interaction with external tools such as search engines, interpreters, and calculators, and its central ablation speaks directly to Section II: with tools, average F1 on question answering improves by 5.6 to 7.7 points over chain-of-thought, while the identical critique loop without tools changes performance marginally or negatively [10]. Critique works exactly insofar as it is externally grounded.

The three architectures differ in where verification bites: execute the reasoning, solve the formalization, or ground the critique. They agree on the premise that there must exist an authoritative substrate against which claims can be checked. How far this paradigm scales is no longer speculative. A neuro-symbolic system in which a language model proposes constructions while a symbolic deduction engine carries and certifies every proof step solved 25 of 30 International Mathematical Olympiad geometry problems, with correctness guaranteed by construction [11]. The question this survey pursues is what plays the role of the interpreter, the solver, and the proof engine when the claims concern roads, lanes, and restrictions. The natural candidate is execution against the map itself.

IV. CONSTRAINED GENERATION: SYNTAX IS NOT TRUTH

A different family of methods intervenes during generation rather than after it: constrain decoding so that invalid outputs cannot be produced at all. SynchroMesh demonstrates the approach for code generation, combining constrained semantic decoding with retrieval of similar examples [12]. The pattern of its gains matters here. On text-to-SQL, validity of the generated queries rises sharply (from 49% to 76% for GPT-3 175B, and to essentially 100% for a visualization language), while end-task accuracy improves far more modestly (28% to 37% execution accuracy on the same model). Finite-state formulations of the idea later made it cheap enough to industrialize [13], and grammar-constrained decoding now ships as the “structured outputs” mode of major inference stacks. Producing syntactically valid output is, as of this writing, a commodity.

Validity, however, is not truth, and the distortion is not merely conceptual. Grammar-aligned decoding proves that standard grammar-constrained decoding samples from a different distribution than the model’s own conditional distribution over grammatical outputs: greedy token masking systematically favors grammatical continuations the model itself considers unlikely [14]. Its corrective algorithm provably converges to the right distribution, yet even the authors report mixed downstream effects. The diagnosis matters more here than the cure: a decoder can guarantee that every output parses while silently degrading what the output says.

The frontier of decode-time guarantees has since moved from syntax toward semantics. Type-constrained code generation enforces well-typedness, a property beyond context-free syntax, soundly during decoding, cutting compilation errors by more than half across six models [15]. This is the strongest decode-time guarantee published to date, and

its boundary defines the open problem that motivates the rest of this survey. Every existing decode-time enforcement targets a closed, decidable theory: a grammar, a schema, or a type system. None targets an external, factual world model. A generated claim about a road network can be perfectly grammatical, perfectly typed, and false. For factual validity, the check must consult the data itself.

V. EXECUTION-BASED EVALUATION

The discipline with the longest experience in checking generated artifacts against data is text-to-query translation. Spider established its modern form: 10,181 questions paired with 5,693 complex SQL queries over 200 databases from 138 domains, evaluated under a cross-database split in which test-time schemas are never seen in training [16]. Spider also exposed the weakness of its own original metric. Exact matching of query structure produces false negatives on semantically equivalent formulations and, because predicted values were not evaluated at all, false positives on queries that match structurally but answer a different question.

The repair of that metric is, for this survey, the single most transferable result in the corpus. Distilled test suites approximate semantic equivalence behaviorally: candidate and gold queries are executed over a compact suite of database instances distilled from a thousand random databases per schema, and the candidate is accepted only if it agrees with the gold query on every instance [17]. The distilled suites distinguish over 99% of mutated “neighbor” queries, where checking against a single database fails to distinguish 5% of them. In a manual audit of 100 disagreements with the old metric, the test-suite verdict was correct in all 100 cases, while adapted exact matching showed a mean false-negative rate of 2.6% (up to 8.1%) across 21 real leaderboard submissions. The principle generalizes far beyond SQL: semantic correctness of a generated claim can be approximated, with measurable false-accept bounds, by execution over a designed family of data instances. Nothing in that construction is specific to relational databases; the same idea transfers, in principle, to road networks.

Execution-graded evaluation has recently reached the data model closest to ours. CypherBench evaluates text-to-Cypher over 11 property graphs materialized from Wikidata, containing 7.8 million entities, scoring predicted queries by execution accuracy against the graph and by the overlap of the provenance subgraphs their queries touch [18]. The results calibrate expectations: the best frontier model reaches only 61.6% execution accuracy, and the best open-weights model 41.9%. Querying large, typed graphs correctly is not a solved problem even before any spatial semantics enter the picture. Geodata itself has one execution-grounded precedent, a natural-language interface to OpenStreetMap evaluated by executing generated queries against the database [19], which we examine in Section VII.

The takeaway of this section is a definition. In the execution paradigm, the correctness of a generated artifact is the outcome of executing it against authoritative data, and the quality of that verdict is an engineering property of the oracle (coverage of

the instance suite, bounds on false accepts) rather than a matter of judgment. The spatial domain possesses richer authoritative data than any database benchmark discussed above. What it lacks, as the next two sections show, is the harness.

VI. GROUNDING IN KNOWLEDGE GRAPHS

Knowledge graphs are the setting where checking claims against a symbolic structure has been developed furthest, and road networks are, formally, attributed graphs, which makes this literature the nearest methodological neighbor of our domain.

One line of work makes reasoning faithful by construction. Reasoning-on-Graphs plans relation paths that are grounded in the knowledge graph, retrieves the corresponding subgraphs, and reasons over them, so that every step of the chain is a checkable path rather than free prose [20]. It reaches 85.7% Hits@1 on WebQSP against 66.8% for an untuned ChatGPT baseline, and its planning module, combined with that same untuned model, lifts it to 81.5%. The template matters more than the scores: constrain the reasoning to artifacts the graph can confirm.

A second line verifies claims rather than constructing reasoning. FactKG poses claim verification directly over a knowledge graph: 108,674 natural-language claims spanning one-hop, conjunctive, existential, multi-hop, and negated forms [21]. Models with access to retrieved graph evidence reach 77.7% accuracy where claim-only baselines reach 65.2%, a 12.5-point contribution from graph access, and multi-hop claims remain the hardest category even with evidence. FactKG is the closest existing analogue of spatial claim checking, and its spatial counterpart does not exist: to our knowledge, no benchmark verifies natural-language claims about turn restrictions, lane connectivity, or temporal validity against a road network.

The third line closes the loop. Knowledge-graph retrofitting extracts the factual claims from a drafted answer, verifies each against the graph, and revises the draft accordingly: detect, verify, repair [22]. On SimpleQuestions, exact match for an unaided language model rises from 22.0% to 58.0%. The caveat is that its evaluations use small samples (50 questions per dataset), so individual numbers carry variance even though the architecture is the point. Alongside these, the broader integration of language models and knowledge graphs has been mapped as a research roadmap [23], and graph-based retrieval augmentation has been industrialized [24]. Both, however, use the graph to inform generation. Retrieval is not verification: an answer assembled from retrieved facts can still compose them into a false claim. The verification pattern of grounding, checking, and repairing exists and works on encyclopedic graphs. For spatial graphs, with their lane topology, temporal attributes, and weekly releases, that carry-over has not happened.

VII. SPATIAL REASONING: CAPABILITY BENCHMARKS, NO VERIFICATION

What exists so far for spatial data is capability measurement, and its results justify concern. MapEval poses 700

map-grounded questions across textual, tool-use, and visual settings: no foundation model exceeds 67% overall, all 30 evaluated models lag human performance by more than 20 points, and accuracy drops further when models must gather context through map APIs themselves (48.7% for GPT-4o on the tool-use split, against 86.7% for humans) [1]. The documented failure modes (distances, cardinal directions, multi-step route planning) are exactly the claim types a navigation assistant emits. Geographic hallucination has likewise been benchmarked directly: on GeoHaluBench, built over a spatial knowledge graph of three cities, the best of 20 evaluated models answers barely half correctly, a random baseline outperforms some open models, and omission errors dominate fabrication [25]. The same work shows the limit of a training-time response. Preference-optimizing a model for geographic factuality lifts average accuracy substantially, but provides, in the authors' framing, no per-answer guarantee, no inference-time checking, and no repair loop.

Two efforts approach verification without reaching it. Text-to-OverpassQL evaluates generated OpenStreetMap queries by executing them against the database, which is the execution paradigm of Section V arriving in geodata, and the absolute numbers are low: GPT-4 reaches an execution accuracy of only 39% on the test set [19]. Two of its details matter for this survey. First, self-refinement driven by execution feedback repairs most syntax errors (from 24 down to 7) but barely moves execution accuracy, while refinement without that feedback repairs far fewer (24 down to 19). The pattern of Section II recurs in the spatial domain: the value lies in the external signal. Second, its execution ground truth is pinned to one frozen 306-gigabyte OSM snapshot, released precisely because results would not be reproducible against the live, continuously edited map. The version problem that Section VIII takes up appears here in embryonic form. More recently, the MapReason-OSM preprint has evaluated vision-language models by snapping their mobility decisions back onto a hidden OSM graph and scoring validity and legality by graph oracle [26]. The verification instinct is right; the study uses synthetic attributes over the road networks of ten city centers and is not yet peer-reviewed.

Table I assembles the comparison. Read by rows, it shows mature ingredients: execution oracles exist (the Spider lineage, CypherBench, Text-to-OverpassQL), claim verification over graphs exists (FactKG), and spatial capability measurement exists (MapEval, GeoHaluBench). Read by columns, it shows the gap this survey documents. No benchmark in any lineage covers lane-level topology, turn restrictions, temporal validity, or map-version provenance, the four dimensions on which navigation claims depend.

This gap has direct industrial consequences. In current industrial practice, production automotive maps are lane-level graphs with temporally conditioned attributes, re-released on a weekly cadence to track a world that changes continuously (Fig. 1), and any claim made against them, whether by a navigation assistant or by the model-based tools that maintain the map, is evaluated against a specific map version that the consumer of the claim may not share. Parts of the machinery already exist in production: routing engines act as executable

TABLE I
BENCHMARKS RELEVANT TO VERIFIABLE SPATIAL REASONING, BY ORACLE TYPE AND SPATIAL COVERAGE.

Benchmark	Data source	Oracle	Spatial	Lanes	Turn restr.	Temporal	Versioned
Spider [16]	200 relational DBs	execution	—	—	—	—	—
CypherBench [18]	Wikidata property graphs	execution	—	—	—	—	—
FactKG [21]	DBpedia knowledge graph	graph evidence	—	—	—	—	—
MapEval [1]	Google Maps APIs	reference answers	✓	—	—	—	—
GeoHaluBench [25]	spatial KG (3 cities)	reference answers	✓	—	—	—	—
Text-to-OverpassQL [19]	OSM (frozen snapshot)	execution	✓	—	—	—	—
MapReason-OSM [26] [†]	OSM (10 cities, synth. attrs.)	graph oracle	✓	—	—	—	—

[†]Preprint, not yet peer-reviewed at the time of writing.

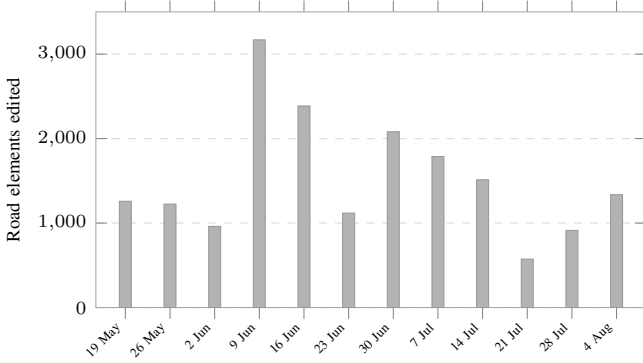


Fig. 1. Road-network elements in the Warsaw area whose latest OpenStreetMap edit falls in each of twelve consecutive weeks (19 May – 11 Aug 2026), measured via the Overpass API. Between 573 and 3,169 road elements changed every single week; the authoritative graph is never static.

oracles for reachability and cost, and map matching binds observations to the graph. What is missing is their use as claim checkers for language-model outputs, with lane-level, temporal, and version coverage. Verification that ignores lanes cannot check lane guidance; verification that ignores time cannot check a rush-hour restriction; verification that ignores versions cannot even say which map it certified. The final section turns this diagnosis into a research agenda.

VIII. RESEARCH AGENDA: FROM CHECKS TO CERTIFICATES

Table II synthesizes the mechanisms reviewed so far along three axes: what is guaranteed, when the mechanism applies, and what evidence it leaves behind. Two patterns organize the field. Reading down the table, guarantees strengthen from syntactic form to factual consistency with authoritative data. In addition, the mechanisms that produce durable evidence, meaning artifacts a third party could re-check, cluster at the bottom. The agenda below extends that bottom row to spatial data.

The first open problem is a specification: validity contracts for spatial claims (Fig. 2). A contract binds an assertion in a defined claim language to the exact map version it was evaluated against, the witness that supports it, and a checker verdict. The claim classes that arise in practice suggest the shape of the language: topological claims (reachability, connectivity), attribute claims (speed limits, access restrictions), temporal

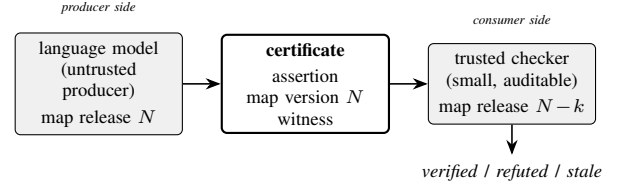


Fig. 2. The proposed certificate pipeline. An untrusted producer evaluates a claim against map release N and emits a certificate binding the assertion to that version and its supporting witness. A small trusted checker on the consumer side re-verifies the certificate against the release it actually holds, $N - k$, and reports one of three verdicts.

claims (time-conditioned validity), lane-level claims (lane connectivity, permitted maneuvers), and provenance claims (which source and version support an attribute). Each class requires an executable oracle over the road-network graph, and the classes plausibly differ in how expensive that oracle is. This trade-off between expressiveness and checkability has, to our knowledge, not been mapped for spatial data. A contract also makes the weakest link explicit: translation. The surveyed evidence shows that turning language into formal queries is itself error-prone (39% execution accuracy on OpenStreetMap queries, 61.6% on property graphs), and a certificate for a mistranslated assertion certifies the wrong claim. The design consequence is that the claim language must be small and readable, so that translation is closer to slot filling than to open semantic parsing, and the assertion itself must travel in the certificate, where its consumer can inspect it. A certificate certifies the assertion; the assertion is auditable against the sentence. We are deliberate about terminology. Such certificates are witness-checkable structures in the tradition of certifying computation, not deductive proofs; the analogy to proof-carrying code [29] is architectural rather than logical, resting on the same division between an untrusted producer and a small trusted checker.

The second open problem is measurement. Section V showed that behavioral test suites give semantic verdicts with bounded false accepts [17], and nothing in that construction is specific to SQL. A spatial analogue would generate claims and their oracles programmatically from a versioned road-network graph, so that the benchmark can be regenerated from every data release. Regeneration changes the economics of evaluation: a benchmark that moves with the world cannot be memorized by models trained on its past, which gives contamination resistance by construction rather than by secrecy.

TABLE II
A TAXONOMY OF VERIFICATION MECHANISMS FOR LANGUAGE-MODEL OUTPUTS.

Mechanism	What is guaranteed	When applied	Evidence produced
Grammar-constrained decoding [12], [13]	syntactic validity	during decoding	none (valid by construction)
Type-constrained decoding [15]	well-typedness (closed theory)	during decoding	none (valid by construction)
Program execution [8]	answer equals program output	per answer	program and its output
Solver-in-the-loop with repair [9]	consistency of the formalization	per answer, iterated	formalization, solver verdict
Tool-grounded critique [10]	agreement with external sources	per answer, iterated	tool outputs
Test-suite execution [17]	behavioral equivalence, bounded false accepts	evaluation time	execution results per instance
Single-instance execution grading [18], [19]	agreement with one authoritative instance	evaluation time	execution result, provenance
Graph-evidence checking [20], [21]	claim entailed by the graph	per claim	witness subgraph or path
Detect-verify-repair [22]	per-claim consistency with the graph	per answer, iterated	claim set with verdicts
Runtime shielding [27], [28]	enforcement of a safety property	at runtime, per action	monitor decision log
Spatial validity certificates (proposed)	claim valid on a stated map version	at runtime, re-checkable by consumer	certificate: assertion, version, witness

A two-tier design, with an open tier built on open map data for reproducibility and a production tier on industrial lane-level data for realism, would let the community and industry calibrate against the same yardstick.

The third open problem is where spatial data departs from everything reviewed above: the authoritative graph changes. Production maps are re-released weekly, and the producer of a claim and its consumer, for instance a vehicle holding an embedded map several releases old, generally do not share a version. A certificate verified against release N therefore has an undefined status on release $N-k$, and defining it is, we argue, the central theoretical question of verifiable spatial reasoning: under what conditions does a verdict transfer across a map difference, when must it be re-established, and how should a checker report the honest third verdict, *stale*, when the difference touches the claim’s support? Runtime verification supplies the vocabulary of monitors and verdicts [28]; shielding shows that runtime enforcement over learned components is practical [27]; and recent work proves soundness for a combined language-model-and-checker system in program verification [30]. Temporal knowledge graphs and incremental query maintenance handle evolving graphs, but they answer what is true now; to our knowledge, none of these lines defines whether a verdict issued against version N remains valid at $N-k$. Making verification version-aware, with certificates that state which version of the world they were true in, is the step this agenda leaves open.

This survey has limits of its own. The corpus was screened primarily by the first author, from an industrial vantage point, which shapes both the selection and the agenda. The venue’s reference budget forced the exclusion of relevant work, among it the BIRD lineage of text-to-SQL benchmarks, faithful-reasoning successors of Reasoning-on-Graphs, and question answering over geospatial knowledge graphs with GeoSPARQL. Sources are English-only, and in a field moving this quickly, preprints that may prove important were excluded by design. The agenda above should be read accordingly: it is one practitioner’s map of the gap, not the only possible one.

IX. CONCLUSION

This survey traced a single argument through five literatures. Language models cannot reliably judge their own outputs, and apparent successes of self-correction dissolve once oracle feedback is removed; external checkers such as interpreters, solvers, and tools restore reliability because their authority lies outside the model. Constrained decoding guarantees the form of an output but not its truth, and the strongest decode-time guarantees published to date enforce closed theories rather than facts about the world. Where facts are at stake, the text-to-query lineage supplies the standard: correctness is the outcome of execution against authoritative data, with oracle quality an engineering property rather than a judgment call. The knowledge-graph literature shows the full detect-verify-repair loop working over symbolic structures at scale.

Against this backdrop, the spatial domain presents a paradox. It owns the richest authoritative data structure of any domain surveyed, a versioned, attributed, lane-level graph of the physical world, yet its benchmarks measure only what models know, and no published method verifies natural-language claims against lane-level, temporally conditioned, or versioned road-network data. The agenda that follows from this diagnosis is concrete: a contract language binding spatial claims to executable oracles and exact map versions; benchmarks regenerated from live map releases, so that evaluation moves with the world; and certificates whose validity is defined across the versions that separate a claim’s producer from its consumer, with the honest verdicts *verified*, *refuted*, and *stale*. Road networks change weekly, and the systems that speak about them are beginning to steer vehicles, answer travelers, and edit the maps themselves. Making their claims checkable is feasible with the machinery this survey assembled, and it is necessary for the step from plausible to provable.

REFERENCES

- [1] M. L. Dihan, M. T. Hassan, M. T. Parvez, M. H. Hasan, M. A. Alam, M. A. Cheema, M. E. Ali, and M. R. Parvez, “MapEval: A map-based evaluation of geo-spatial reasoning in foundation models,” in *Proc. 42nd Int. Conf. on Machine Learning (ICML), PMLR 267*, 2025, pp. 13 774–13 813. [Online]. Available: <https://arxiv.org/abs/2501.00316>

- [2] G.-I. Dorobantu and A.-C. Badea, “Geospatial reasoning and awareness in large language models: A systematic review,” *Artificial Intelligence Review*, vol. 59, no. 4, 2026, art. no. 111. [Online]. Available: <https://doi.org/10.1007/s10462-026-11512-x>
- [3] J. Huang, X. Chen, S. Mishra, H. S. Zheng, A. W. Yu, X. Song, and D. Zhou, “Large language models cannot self-correct reasoning yet,” in *Proc. 12th Int. Conf. on Learning Representations (ICLR)*, 2024. [Online]. Available: <https://openreview.net/forum?id=IkmdD3fKBQ>
- [4] K. Stechly, K. Valmeekam, and S. Kambhampati, “On the self-verification limitations of large language models on reasoning and planning tasks,” in *Proc. 13th Int. Conf. on Learning Representations (ICLR)*, 2025. [Online]. Available: <https://arxiv.org/abs/2402.08115>
- [5] A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhume, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh, and P. Clark, “Self-refine: Iterative refinement with self-feedback,” in *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023. [Online]. Available: <https://arxiv.org/abs/2303.17651>
- [6] R. Kamoi, Y. Zhang, N. Zhang, J. Han, and R. Zhang, “When can LLMs actually correct their own mistakes? A critical survey of self-correction of LLMs,” *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 1417–1440, 2024. [Online]. Available: https://doi.org/10.1162/tac1_a_00713
- [7] H. Lightman, V. Kosaraju, Y. Burda, H. Edwards, B. Baker, T. Lee, J. Leike, J. Schulman, I. Sutskever, and K. Cobbe, “Let’s verify step by step,” in *Proc. 12th Int. Conf. on Learning Representations (ICLR)*, 2024. [Online]. Available: <https://arxiv.org/abs/2305.20050>
- [8] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig, “PAL: Program-aided language models,” in *Proc. 40th Int. Conf. on Machine Learning (ICML)*, PMLR 202, 2023, pp. 10 764–10 799. [Online]. Available: <https://proceedings.mlr.press/v202/gao23f.html>
- [9] L. Pan, A. Albalak, X. Wang, and W. Y. Wang, “Logic-LM: Empowering large language models with symbolic solvers for faithful logical reasoning,” in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 3806–3824. [Online]. Available: <https://doi.org/10.18653/v1/2023.findings-emnlp.248>
- [10] Z. Gou, Z. Shao, Y. Gong, Y. Shen, Y. Yang, N. Duan, and W. Chen, “CRITIC: Large language models can self-correct with tool-interactive critiquing,” in *Proc. 12th Int. Conf. on Learning Representations (ICLR)*, 2024. [Online]. Available: <https://openreview.net/forum?id=Sx038qxjek>
- [11] T. H. Trinh, Y. Wu, Q. V. Le, H. He, and T. Luong, “Solving olympiad geometry without human demonstrations,” *Nature*, vol. 625, no. 7995, pp. 476–482, 2024. [Online]. Available: <https://doi.org/10.1038/s41586-023-06747-5>
- [12] G. Poesia, A. Polozov, V. Le, A. Tiwari, G. Soares, C. Meek, and S. Gulwani, “SynchroMesh: Reliable code generation from pre-trained language models,” in *Proc. 10th Int. Conf. on Learning Representations (ICLR)*, 2022. [Online]. Available: <https://arxiv.org/abs/2201.11227>
- [13] B. T. Willard and R. Louf, “Efficient guided generation for large language models,” arXiv:2307.09702, 2023. [Online]. Available: <https://arxiv.org/abs/2307.09702>
- [14] K. Park, J. Wang, T. Berg-Kirkpatrick, N. Polikarpova, and L. D’Antoni, “Grammar-aligned decoding,” in *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024. [Online]. Available: <https://arxiv.org/abs/2405.21047>
- [15] N. Mündler, J. He, H. Wang, K. Sen, D. Song, and M. Vechev, “Type-constrained code generation with language models,” *Proc. ACM Program. Lang. (PLDI)*, vol. 9, pp. 601–626, 2025. [Online]. Available: <https://doi.org/10.1145/3729274>
- [16] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, “Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task,” in *Proc. 2018 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2018, pp. 3911–3921. [Online]. Available: <https://doi.org/10.18653/v1/D18-1425>
- [17] R. Zhong, T. Yu, and D. Klein, “Semantic evaluation for text-to-SQL with distilled test suites,” in *Proc. 2020 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 396–411. [Online]. Available: <https://doi.org/10.18653/v1/2020.emnlp-main.29>
- [18] Y. Feng, S. Papicchio, and S. Rahman, “CypherBench: Towards precise retrieval over full-scale modern knowledge graphs in the LLM era,” in *Proc. 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025, pp. 8934–8958. [Online]. Available: <https://doi.org/10.18653/v1/2025.acl-long.438>
- [19] M. Staniek, R. Schumann, M. Züfle, and S. Riezler, “Text-to-OverpassQL: A natural language interface for complex geodata querying of OpenStreetMap,” *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 562–575, 2024. [Online]. Available: https://doi.org/10.1162/tac1_a_00654
- [20] L. Luo, Y.-F. Li, G. Haffari, and S. Pan, “Reasoning on graphs: Faithful and interpretable large language model reasoning,” in *Proc. 12th Int. Conf. on Learning Representations (ICLR)*, 2024. [Online]. Available: <https://arxiv.org/abs/2310.01061>
- [21] J. Kim, S. Park, Y. Kwon, Y. Jo, J. Thorne, and E. Choi, “FactKG: Fact verification via reasoning on knowledge graphs,” in *Proc. 61st Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023, pp. 16 190–16 206. [Online]. Available: <https://doi.org/10.18653/v1/2023.acl-long.895>
- [22] X. Guan, Y. Liu, H. Lin, Y. Lu, B. He, X. Han, and L. Sun, “Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting,” in *Proc. 38th AAAI Conf. on Artificial Intelligence*, vol. 38, 2024, pp. 18 126–18 134. [Online]. Available: <https://doi.org/10.1609/aaai.v38i16.29770>
- [23] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, “Unifying large language models and knowledge graphs: A roadmap,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3580–3599, 2024. [Online]. Available: <https://doi.org/10.1109/TKDE.2024.3352100>
- [24] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, and J. Larson, “From local to global: A graph RAG approach to query-focused summarization,” arXiv:2404.16130, 2024. [Online]. Available: <https://arxiv.org/abs/2404.16130>
- [25] S. Wang, J. Feng, T. Liu, D. Pei, and Y. Li, “Mitigating geospatial knowledge hallucination in large language models: Benchmarking and dynamic factuality aligning,” in *Findings of the Association for Computational Linguistics: EMNLP 2025*, 2025, pp. 870–888. [Online]. Available: <https://doi.org/10.18653/v1/2025.findings-emnlp.45>
- [26] S. Venkatanarayanan and C. P. Isaac, “MapReason-OSM: Can vision-language models make graph-verifiable mobility decisions from street maps?” arXiv:2606.22597, 2026. [Online]. Available: <https://arxiv.org/abs/2606.22597>
- [27] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, “Safe reinforcement learning via shielding,” in *Proc. 32nd AAAI Conf. on Artificial Intelligence*, vol. 32, 2018. [Online]. Available: <https://doi.org/10.1609/aaai.v32i1.11797>
- [28] M. Leucker and C. Schallhart, “A brief account of runtime verification,” *Journal of Logic and Algebraic Programming*, vol. 78, no. 5, pp. 293–303, 2009. [Online]. Available: <https://doi.org/10.1016/j.jlap.2008.08.004>
- [29] G. C. Necula, “Proof-carrying code,” in *Proc. 24th ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages (POPL)*, 1997, pp. 106–119. [Online]. Available: <https://doi.org/10.1145/263699.263712>
- [30] H. Wu, C. Barrett, and N. Narodytska, “Lemur: Integrating large language models in automated program verification,” in *Proc. 12th Int. Conf. on Learning Representations (ICLR)*, 2024. [Online]. Available: <https://arxiv.org/abs/2310.04870>