Preprint Cambridge Centre for Computational Chemical Engineering ISSN 1473 – 4273

Ontology-to-tools compilation for executable semantic constraint enforcement in LLM agents

Xiaochi Zhou[1], Patrick Butler[1], Changxuan Yang[3], Simon Rihm[5], Thitikarn Angkanaporn[1], Jethro Akroyd[1,2,4], Sebastian Mosbach[1,2,4], Markus Kraft[1,2,3,5]

released: February 6, 2026

[1] Department of Chemical Engineering and Biotechnology
University of Cambridge
Philippa Fawcett Drive
Cambridge, CB3 0AS
United Kingdom

[2] CARES
Cambridge Centre for Advanced Research and Education in Singapore
1 Create Way
CREATE Tower, #05-05
Singapore, 138602

[3] MIT, Chemical Engineering
77 Massachusetts Avenue, Room E17-504
Cambridge, MA 02139 USA

[4] CMCL
No. 9, Journey Campus
Castle Park
Cambridge
CB3 0AX
United Kingdom

[5] CMPG
GRIPS – Gründerinnenzentrum Pirmasens
Delaware Avenue 1–3
66953 Pirmasens
Germany

Preprint No. 343

Keywords: Large language models; autonomous agents; knowledge graphs; scientific information extraction

Edited by
Computational Modelling Group
Department of Chemical Engineering and Biotechnology
University of Cambridge
Philippa Fawcett Drive
Cambridge, CB3 0AS
United Kingdom

E-Mail: mk306@cam.ac.uk
World Wide Web: https://como.ceb.cam.ac.uk/

Abstract

We introduce *ontology-to-tools compilation* as a proof-of-principle mechanism for coupling large language models (LLMs) with formal domain knowledge. Within *The World Avatar* (TWA), ontological specifications are compiled into executable tool interfaces that LLM-based agents must use to create and modify knowledge graph instances, enforcing semantic constraints during generation rather than through post-hoc validation. Extending TWA's semantic agent composition framework, the Model Context Protocol (MCP) and associated agents are integral components of the knowledge graph ecosystem, enabling structured interaction between generative models, symbolic constraints, and external resources. An agent-based workflow translates ontologies into ontology-aware tools and iteratively applies them to extract, validate, and repair structured knowledge from unstructured scientific text. Using metal–organic polyhedra synthesis literature as an illustrative case, we show how executable ontological semantics can guide LLM behaviour and reduce manual schema and prompt engineering, establishing a general paradigm for embedding formal knowledge into generative systems.

Highlights

- Ontological constraints are compiled into executable tools for LLM agents within The World Avatar.
- Tool-using LLM agents iteratively extract and instantiate knowledge under semantic constraints.
- A synthesis literature case study in The World Avatar demonstrates rule-consistent, stateful generation.

Contents

1 Introduction

Extracting structured data from scientific literature is a critical and routine step for enabling machine-actionable scientific workflows, including hypothesis generation, data-driven modelling, and synthesis prediction. [3, 9, 10, 18, 24, 30, 61, 66]. In reticular chemistry, for example, literature extraction can support synthesis predictions of novel materials and estimation of properties such as gas adsorption, and provide means to identify gaps in the immediate chemical space [5, 23, 29, 56, 78, 79]. Moreover, structuring literature makes it possible to link extracted records to complementary data sources, expanding their context and increasing their value beyond what is reported in any single paper [33, 38].

However, turning scientific text into machine-actionable data remains challenging. Scientific text requires precise interpretation and is domain-specific, while downstream uses, such as data-driven modelling, require consistently defined records that are comprehensively validated [18, 30, 55, 66]. These records need clear roles and concept boundaries, normalized units and values, and, crucially, unambiguous entity references [38, 55, 64]. For example, a reagent may be mentioned without an explicit role (precursor *vs* solvent), experimental conditions may be expressed in inconsistent unit forms (*e.g.* C *vs* Degrees Celsius), and the same chemical may appear under multiple names or abbreviations; downstream workflows therefore require canonical identifiers (*e.g.*, CAS numbers or InChIKeys) and consistent typing to support validation, deduplication, and cross-paper integration [46, 51].

Ontologies are one natural option for specifying these roles, boundaries, and constraints: an ontology is a formal, machine-readable description of a domain, typically decomposed into a T-Box that defines concepts and relations, and an A-Box that instantiates them with concrete entities and facts [19, 20]. In such knowledge graphs, entities are represented by Internationalized Resource Identifiers (IRIs), which serve as unique identifiers for individual instances, such as specific chemical species [60]. In parallel, recent large language models (LLMs) have enabled few-shot and zero-shot adaptation for extraction tasks, and can apply prompt-based, soft definitions of roles and boundaries by interpreting context and producing schema-shaped records [22, 36, 66, 70, 78].

Pipelines that aim to produce structured data ready for downstream use often combine ontologies (to define target roles, relations, and constraints) with extractors (LLMs or otherwise) to interpret text. However, downstream use typically requires that extracted records conform to the ontology, including valid types and relations, normalized values and units, and grounded identifiers. This concentrates effort in constraint enforcement, such as validation, normalization, and grounding or alignment, implemented within the extraction system or as a separate step [38, 64, 66]. In non-LLM pipelines, this enforcement is commonly realized through domain-specific components such as trained models, hand-engineered rules, and ontology-aligned assembly procedures [38, 64].

Many other approaches shift this work to post-hoc processing: they first generate schema-shaped records, then apply deterministic validation and ontology/database alignment to ensure the outputs satisfy required fields, normalization, cross-field consistency, and identifier grounding [37, 55, 66, 67]. As a result, achieving ontology-conformant data often

depends on hand-built validation/alignment logic outside the extractor, rather than being enforced during extraction itself.

Across these families, the common limitation is how downstream requirements are enforced: they are implemented either as bespoke, domain-specific pipeline logic around extraction or as a separate post-hoc layer for validation, normalization, and grounding and alignment, concentrating expert effort in code that must be revised as schemas and scope change.

Recently, large language models (LLMs) and LLM-agents have increasingly supported tool calling, where a model invokes external functions (scripts and APIs) during generation to produce structured outputs, perform deterministic checks, and retrieve supporting evidence. Frameworks such as the Model Context Protocol (MCP) standardize this interaction by providing a common interface for registering tools and exchanging typed inputs and outputs [1, 32, 33, 37, 40, 43, 50, 54, 65, 69]. These capabilities make it possible to enforce structure and domain constraints during extraction, rather than relying solely on post hoc validation.

Such constraints are particularly important in scientific domains, where extracted information must align with well-defined ontologies and existing knowledge bases. The World Avatar [5] is a large-scale dynamic knowledge graph for chemistry that provides curated T-Boxes and extensive A-Box instances covering synthesis descriptions, metal organic polyhedra, and chemical species [29, 46, 51]. As a result, it offers a natural grounding target for tool-augmented LLM extraction, supplying ready-to-use semantic schemas against which generated outputs can be validated and instantiated.

Motivated by the constraint-enforcement bottleneck identified above, we introduce a novel ontology-to-tools compilation framework that turns an ontology into executable LLM-callable tools, enabling constraints to be enforced during extraction rather than by bespoke pipeline logic or post-hoc validation. To our knowledge, no prior system has transformed an ontology into an LLM-executable constraint enforcement layer. In our approach, an LLM-driven agent performs document-level extraction and invoking compiled tools to construct the knowledge graph directly: tool calls instantiate individuals and relations with built-in constraint checks and structured feedback on violations. The same compilation framework also yields lexical grounding tools. These tools link surface text mentions to ontology-defined entities using lexical labels and evidence from a reference knowledge graph. In practice, they align extracted mentions to the IRIs of existing instances in the graph, for example an already recorded chemical species.

Our contribution is: (1) an ontology-to-tools compilation framework for generating ontology-aligned prompts and MCP tools from a *T-Box* and meta-prompts; (2) an ontology-constrained KG construction procedure that enforces constraints at creation time via tool calls with structured feedback; and (3) an ontology-driven grounding workflow that generates lexical grounding tools from the *T-Box* and endpoint evidence for identifier alignment. We demonstrate the ontology-to-tools compilation framework on metal–organic polyhedra (MOP) synthesis literature in The World Avatar, covering ontology-constrained knowledge-graph construction from synthesis text, followed by lexical grounding to canonical identifiers.

From a machine-intelligence perspective, the central contribution is a compilation mecha-

nism that transforms symbolic constraints into executable action interfaces for generative models. This reframes constraint enforcement from prompt- or schema-based output constraints and post-hoc validation into run-time interaction with a structured environment. The resulting behaviour is not achieved through prompt engineering or grammar restriction, but through tool-mediated interaction with a persistent symbolic state. The chemistry case study serves as a concrete instantiation of this general mechanism.

2 Knowledge-Graph Construction from Metal–Organic Polyhedra Synthesis Literature

We demonstrate the ontology-to-tool compilation framework by constructing grounded, ontology-consistent knowledge graphs from metal–organic polyhedra synthesis literature. Figure 1 illustrates the core knowledge-graph representation produced for metal–organic polyhedra (MOP) synthesis papers. The results reported in this section are obtained from 30 MOP-related publications, with each paper treated as a full-text input including manuscript text, tables, and supplementary information when available.

The task definition is to convert unstructured MOP synthesis literature into grounded, ontology-consistent knowledge-graph instances. Given a full-text synthesis paper, the output is a set of interlinked instances capturing synthesis procedures, chemical reagents, and the reported MOP, with extracted entities linked to existing knowledge graph instances.

The representation draws on multiple complementary domain ontologies, including OntoSynthesis [51] for modelling synthesis events, ordered steps, conditions, and reagent usage; OntoSpecies [46] for canonical chemical species identities and identifiers; OntoMOPs [29] for MOP products, structural concepts, and derived chemical building units (CBUs); and OM-2 for representing and normalizing quantities and units in synthesis descriptions. Chemical species are grounded at the usage-instance level by linking extracted instances to canonical OntoSpecies entries.

This task is challenging because it requires the coordinated use and alignment of multiple ontologies when analysing a single paper. Procedural knowledge, canonical chemical identity, and product- and structure-level material knowledge, including derived CBUs, must be populated consistently and linked across ontology boundaries. Moreover, the relevant information is often reported at different levels of detail, and the resulting knowledge graph must keep track of where each piece of information comes from, so that roles, amounts, and other qualifiers remain correctly associated with the appropriate chemical usage and synthesis steps. In addition, CBU derivation requires connecting extracted synthesis evidence to product representations and external chemical or crystallographic data while maintaining consistent identifiers across all ontological layers.

For each paper, the framework produces three interconnected outputs. First, a structured synthesis recipe captures ordered synthesis steps, step-level actions, and associated conditions, together with additional synthesis-related details such as reagent usage and provenance information.

Second, a set of grounded chemical species instances represents chemical entities uniquely,

with contextual qualifiers such as role, amount, and units attached to each occurrence in the synthesis, and with links to canonical OntoSpecies records for cross-paper integration.

Third, derived chemical building units (CBUs), including reusable metal nodes or clusters and organic ligands, are obtained by combining extracted evidence with database-backed chemical and crystallographic data. These CBUs are instantiated under OntoMOPs and linked back to the reported MOP product and the synthesis in which they were produced.

Figure 1 shows an example ontology instance produced for a representative synthesis. Panel (A) presents an excerpt of the instances subgraph, *i.e.* A-Box, including a synthesis event linked to an ordered sequence of steps, step-scoped reagent usage, and the reported MOP product, together with attributes such as yield and representation links. Panel (B) shows a compact, record-style projection of the same instances in canonical slot–value form used for inspection and downstream querying. In this representation, contextual qualifiers remain attached to individual usage instances, while selected inputs are grounded to canonical OntoSpecies entries via identity links. Together, the outputs integrate procedural structure, species grounding, and product- and CBU-level semantics to yield a single, queryable knowledge-graph instance.

3 Design of Ontology-to-Tools Compilation Framework

This section describes how the ontology-consistent knowledge-graph instances defined in Section 2 are constructed from unstructured documents. The focus here is on the compilation and execution mechanisms that translate ontology specifications into executable tools and use them to perform constrained extraction. Figure 2 provides an overview of the end-to-end workflow. The compiled tool interfaces collectively define the action space available to the LLM agent, constraining generation through executable semantics rather prompt- or schema-based constraints on the output.

As illustrated in Figure 2, the framework has two stages: (1) the *preparation stage*, which compiles an ontology into tools and prompts, and (2) the *instantiation stage*, which runs a tool-using agent to construct the knowledge graph from papers. In the *preparation stage*, the preparation agent consumes an ontology schema (T-Box) together with domain-agnostic meta-prompts (instruction templates) and generates ontology-aligned runtime prompts, supporting scripts, and LLM-callable tool interfaces exposed via the Model Context Protocol (MCP). These tools implement ontology-aware instance construction with built-in validation logic. In the *instantiation stage*, the instantiation agent applies the generated prompts, scripts, and LLM-callable tools to each document, invoking tools to create and link individuals and relations; constraint violations are returned as structured feedback to support iterative completion and repair.

The compilation layer treats the T-Box as a machine-readable contract. It specifies which classes, relations, attributes, and constraints are allowed. From this contract, the framework generates executable tool interfaces with explicitly specified inputs, outputs, and validation behaviour. These tools are the only way to create or modify structured instances, so constraints are checked and repaired during construction rather than only after the fact.

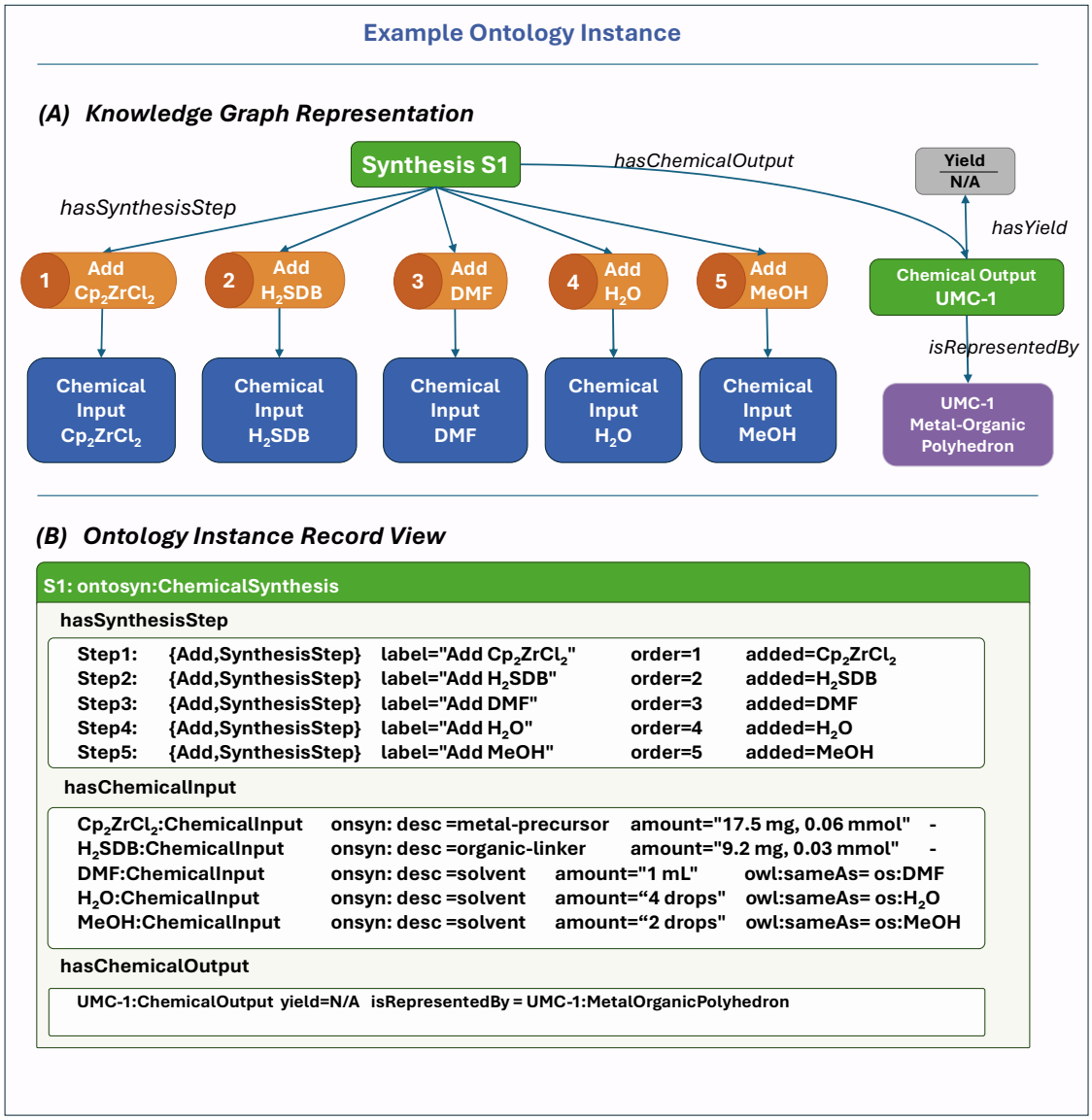

Figure 1: ***Example ontology instance produced by the instantiation agent.*** *(A) A-Box subgraph instantiated under OntoSynthesis/OntoMOPs for synthesis* `S1`*, including ordered synthesis steps, chemical inputs, and product* `UMC-1` *(with yield and representation links). (B) Compact record projection (A-Box normal form) of the same instance in canonical slot–value form;* `onsyn:*` *abbreviates* `ontosyn:*`*, and selected* `ChemicalInput` *entities are grounded via* `owl:sameAs` *to* `ontospecies:Species`*.*

During instantiation, the extraction agent interprets document content and issues tool calls to construct synthesis steps, usage instances, species links, and product entities. External chemical and crystallographic resources are integrated as callable tools within the same execution framework, enabling canonical species identification and database-backed derivation of higher-level entities such as CBUs. All extracted, grounded, and derived entities are instantiated into a knowledge graph under the constraints defined by the ontology contract. Together, these components implement an end-to-end workflow in which ontology specifications and unstructured literature jointly drive the construction of grounded, internally consistent knowledge-graph instances, without relying on post-hoc validation or manual schema-specific repair.

4 Performance of Ontology-to-tool compilation

This section evaluates the ontology-to-tool compilation framework and the resulting knowledge-graph construction on the same curated corpus of 30 metal–organic polyhedra (MOP) synthesis articles used throughout this work. Two questions matter. First, are the generated graphs semantically healthy, *i.e.* do they satisfy the ontology constraints and remain structurally consistent under instantiation? Second, is the extracted and grounded content accurate with respect to the source literature?

To answer these questions, we compare the generated outputs against manual ground-truth annotations in predefined JSON record formats covering four target categories: grounded and derived CBUs, characterisation entities, synthesis steps, and reaction chemicals. Section 6.4 describes the dataset in detail. We report *graph-recoverable* performance first. This measures whether the information required by the evaluation schema is actually present in the constructed knowledge graph in the expected ontology form, *i.e.* as individuals and relations that can be retrieved deterministically. Concretely, for each target JSON record type, we use a fixed SPARQL query to reconstruct the record from the graph, and we score the reconstructed records against the ground truth. SPARQL querying provides the recoverability test because it requires the relevant facts to be encoded as explicit triples with the correct links between entities, rather than appearing only as free text, partial attributes, or disconnected nodes. The SPARQL queries are fixed in advance and are not adapted per paper or per predicted graph. They are derived from the target ontological schema (T-Box) and the corresponding JSON record definitions, and are written once to specify how each record should be read out from any valid instantiation. As a result, a prediction is counted as graph-recoverable only if it can be reconstructed through these schema-derived queries. Figure 3 summarises aggregate performance, class imbalance, and per-paper variability. Figure 4 isolates the effects of external grounding services and constraint feedback. Figure 5 analyses dominant step-level error sources and highlights priorities. Exact scores and ablations appear in Tables 7.2 and 7.3. Our evaluation is designed to assess interaction-based constraint enforcement by analysing its effects on graph recoverability and accuracy.

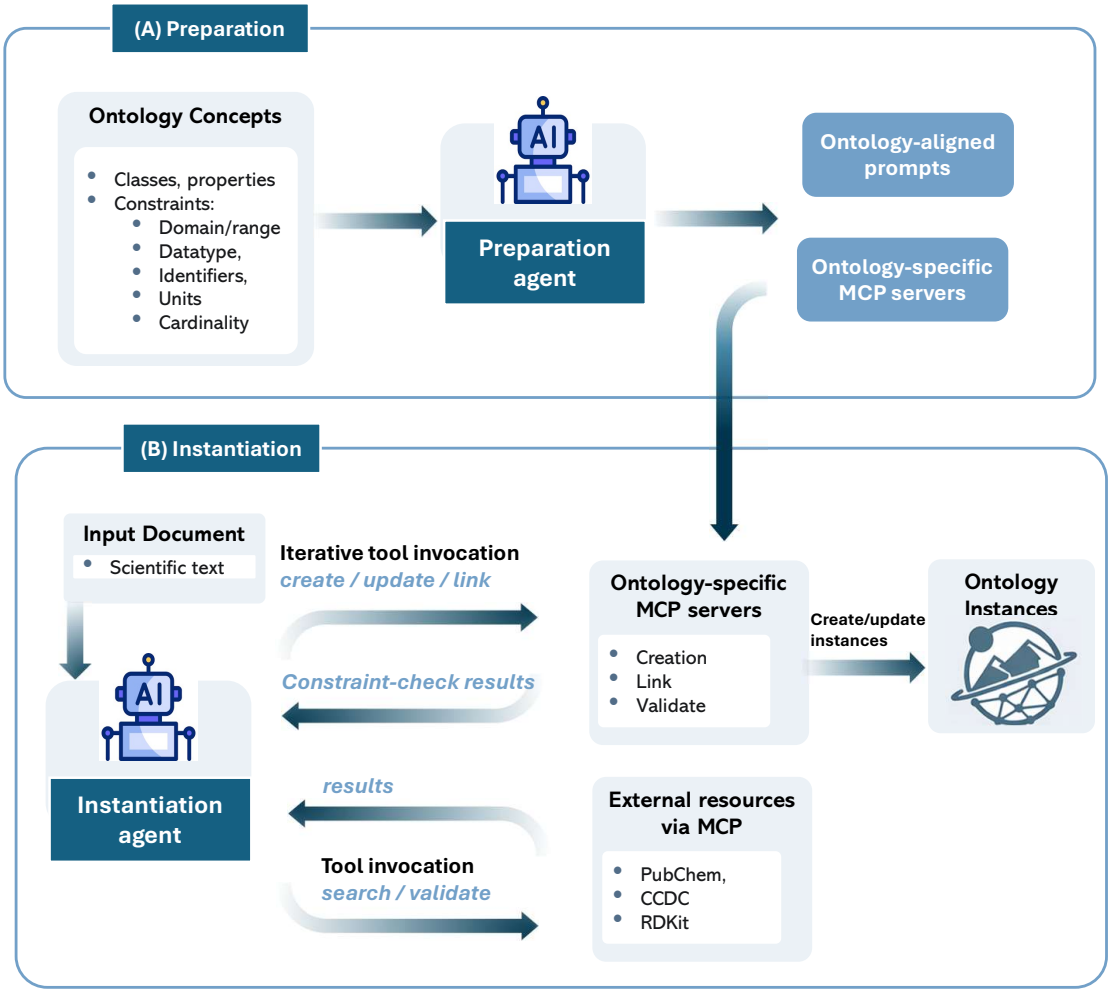

Figure 2: *Ontology-to-tools compilation as an executable semantic control layer for LLM-based agents. Symbolic ontological definitions (T-Box) within The World Avatar are compiled into executable tool interfaces and validators that define the action space available to a large language model during generation. Rather than producing free-form text, the LLM interacts with a persistent symbolic state by invoking ontology-aligned actions that create, modify, and validate graph instances. Constraint violations trigger structured feedback, enabling iterative repair and grounding to external resources. This reframes semantic constraint enforcement from post-hoc validation or constrained decoding into run-time interaction with an evolving symbolic environment, allowing the model to operate as a stateful, ontology-aware agent.*

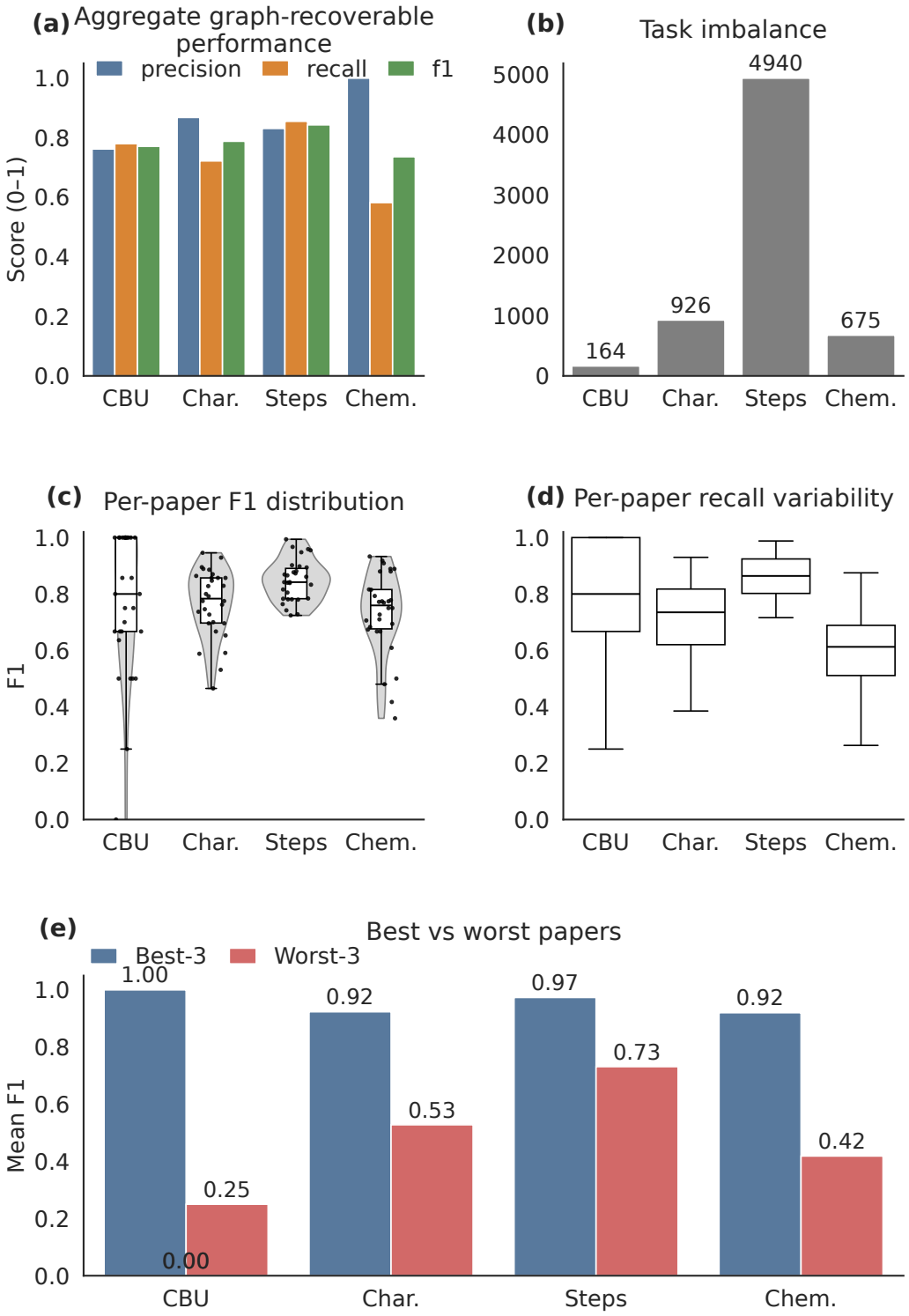

Figure 3: ***Core end-to-end results across four extraction/instantiation domains.*** *(a) Aggregate graph-recoverable precision, recall and F1 for grounded/derived CBUs, characterisation entities, synthesis steps and reaction chemicals (Table 7.2). (b) Task imbalance in ground-truth positives (Table 7.1). (c) Per-paper F1 distributions across the 30-paper benchmark. (d) Per-paper recall variability, highlighting recall-limited categories. (e) Best–worst paper contrast (mean F1 over top-3 vs bottom-3 papers) summarising dataset heterogeneity.*

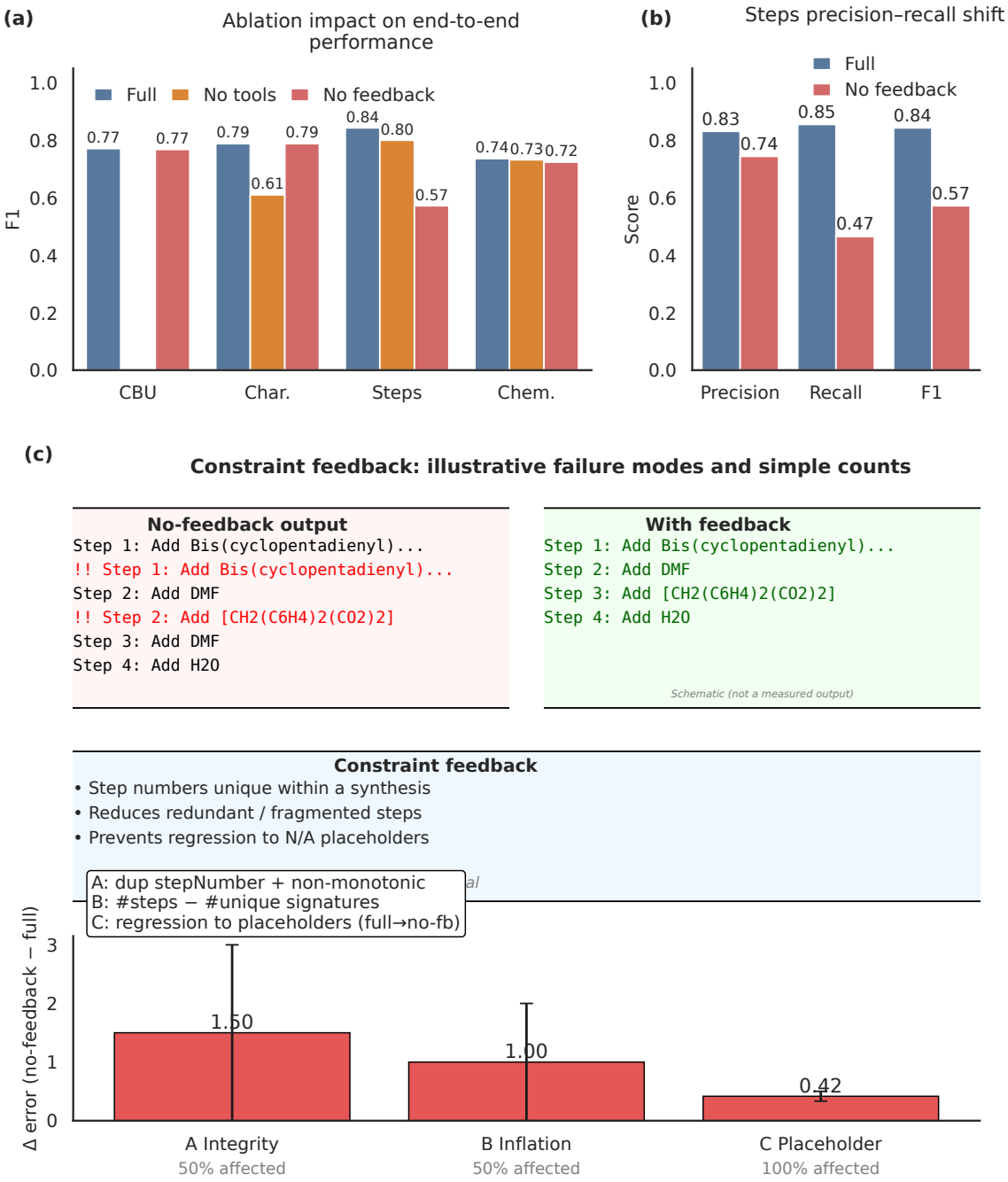

Figure 4: ***Component necessity and the role of constraint feedback.*** *(a) Ablation impact on end-to-end F1 by category (Table 7.3). (b) Steps-only precision–recall–F1 shift under feedback removal, illustrating the dominant effect on step completeness/recoverability. (c) Constraint feedback: illustrative failure modes and paired Δ error counts computed over aligned full vs no-feedback syntheses. The excerpt shows typical no-feedback degradations (e.g. step-number integrity and redundancy), while the mini-plot quantifies three paired error proxies (A–C; defined in-panel) with bootstrap confidence intervals.*

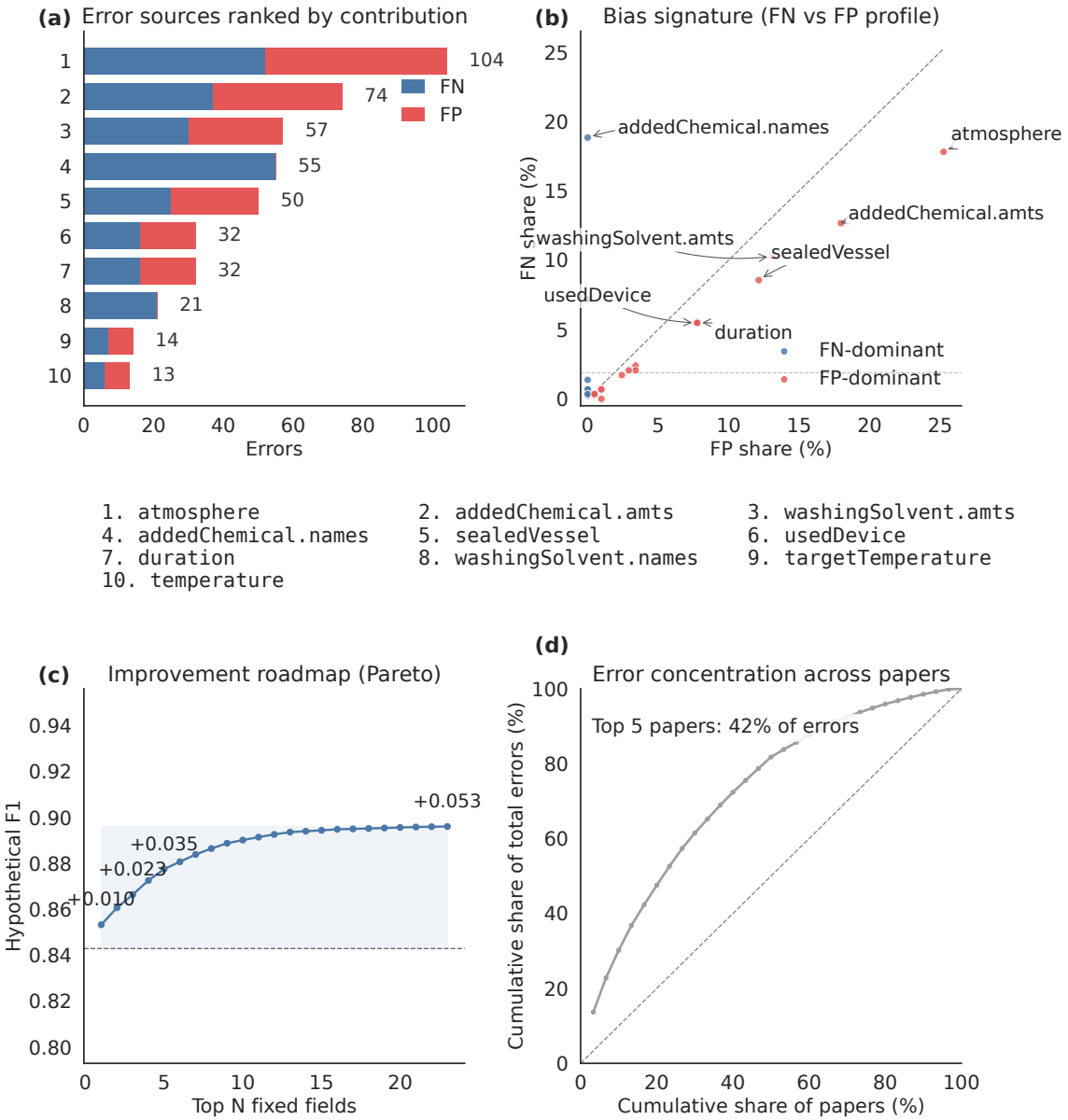

Figure 5: ***Error anatomy and improvement priorities for synthesis steps.*** *(a) Top error-contributing fields (FP vs FN) aggregated over the benchmark. (b) Field-level bias signature (FN-share vs FP-share), separating recall-limited from precision-limited fields. (c) Hypothetical improvement roadmap (Pareto): cumulative F1 if the top-N error-contributing fields were corrected. (d) Error concentration across papers (Lorenz-style curve), showing whether a small subset of papers accounts for a disproportionate share of step errors.*

4.1 Ontology-compiled MCP tools enable semantically valid and content-correct end-to-end graph instantiation

To assess whether ontology-compiled MCP tools produce outputs that are both semantically valid under the ontology and content-correct, we evaluate end-to-end graph-recoverable performance and summarise the results in Figure 3. Figure 3a shows strong overall performance (micro-F1 0.826; Table 7.2), with high precision (0.844) and task-dependent recall (0.808). It also shows clear differences across categories. Synthesis steps perform best overall (F1 0.843). Reaction chemicals have perfect precision (1.000) but much lower recall (0.582). This suggests the system often avoids uncertain chemical mentions and misses some valid ones, especially when names are written in inconsistent forms (Table 7.2).

These scores reflect both semantic and content correctness. A prediction counts as correct only if it is instantiated with the right entity types, relations, and required fields. It must also be recoverable by fixed SPARQL queries that reconstruct the expected JSON records. Errors in either content or structure, including missing individuals, wrong types, missing or incorrect links, or unfilled required slots, prevent recovery and are counted as false negatives.

Figure 3b quantifies the benchmark imbalance. This explains why micro-aggregates are dominated by high-volume categories and motivates reporting macro-averaged scores (macro precision 0.865, macro recall 0.735, macro F1 0.785; Table 7.2). Figure 3c shows broad per-paper F1 distributions across all categories. Figure 3d indicates that recall variability drives the lower tail for some domains. Figure 3 further shows that performance varies widely across papers. The best–worst contrasts suggest that differences in how authors report experiments, and how much concrete detail they include, have a strong impact on what the system can reliably extract and reconstruct from the graph.

In all, the results indicate that ontology-compiled MCP tools support semantically valid graph construction with strong content accuracy, while remaining errors are concentrated in recall-sensitive categories and in papers with sparse or unevenly reported evidence.

4.2 Constraint feedback improves completeness of instantiated synthesis steps

To assess the contribution and necessity of constraint feedback, we ablate the feedback channel and compare end-to-end instantiation scores. Figure 4a summarises the category-level impact of this ablation and shows that synthesis steps are among the most affected outputs. We therefore analyse synthesis-step behaviour in more detail.

In the ablated setting, we disable ontology-derived validation feedback at run time. We manually modify the AI-generated execution script and comment out the code paths that return feedback to the agent. This removes checks for required fields, unit and value normalization, step ordering and continuity, and other consistency constraints that are otherwise applied incrementally as synthesis steps are constructed. The agent therefore generates outputs without receiving signals about missing fields, invalid values, or struc-

tural violations.

Table 7.3 shows a substantial drop in synthesis-step F1 when constraint feedback is removed, driven primarily by lower recall rather than precision. Figure 4b visualises this shift, showing that without feedback the system produces fewer synthesis-step instances that are complete enough to be recovered by the fixed SPARQL evaluation queries.

Figure 4c helps explain the underlying failure modes. Without feedback, step outputs more frequently violate basic structural expectations, including inconsistent or missing step numbers and unnecessary fragmentation into extra steps. The figure reports paired differences between full and no-feedback runs on the same papers using three proxy measures computed from the instantiated outputs. These capture increases in step-number inconsistencies, inflation in the number of steps, and regressions to placeholder or default values. Together, these results show that constraint feedback is important for guiding the model toward synthesis-step structures that are complete, well-formed, and reliably instantiable.

4.3 Dominant synthesis-step error modes and improvement priorities

To identify the main sources of remaining limitations in synthesis-step extraction and to guide future improvements, we analyse synthesis-step errors in detail using Figure 5. Synthesis steps are the core output of the pipeline because they represent the ordered experimental procedure and link together materials, conditions, and outcomes. Errors at this level therefore have a direct impact on the usefulness of the extracted knowledge graph, which motivates a more detailed analysis than for the other categories.

Figure 5a shows that synthesis-step errors are not evenly distributed across fields. A small number of fields account for a large share of the total errors. These include fields that encode step order, action descriptions, and key parameters. Errors arise both as missing fields, where required information is not extracted or instantiated, and as extra fields, where values are produced but do not correspond to the ground truth. This concentration indicates that overall performance is limited by a small set of recurring failure points rather than uniform noise across the schema.

Figure 5b further separates fields by error type. Some fields are recall-limited, meaning the system often fails to extract values that are present in the paper. Other fields are precision-limited, meaning the system more often produces incorrect or unnecessary values. This distinction suggests different improvement strategies: recall-limited fields benefit from better coverage and normalization, while precision-limited fields require tighter constraints and filtering.

Building on this ranking, Figure 5c presents an improvement roadmap under a conservative fix-by-field assumption. It estimates how synthesis-step F1 would increase if the highest-impact fields were corrected one at a time. The curve shows diminishing returns, where correcting only a few top contributors yields a large fraction of the achievable improvement.

Finally, Figure 5d shows that synthesis-step errors are unevenly distributed across pa-

pers. A minority of papers accounts for a large fraction of the total errors, suggesting that targeted analysis of the most difficult papers can complement global field-level improvements.

5 Discussion

This work demonstrates a minimal constructive example of how generative models can be coupled to formal systems through executable semantics. Rather than treating symbolic knowledge as static context or validation targets, ontologies are operationalised as run-time constraints that shape agent behaviour through interaction. This shifts the role of large language models from text generators to stateful programs operating within a structured environment. More broadly, ontology-to-tools compilation suggests a pathway for operationalising symbolic constraints as run-time, feedback-driven mechanisms that shape model behaviour, rather than encoding them only as static representations.

5.1 Limitations

The current evaluation covers one ontology and one document collection. We use a MOP-focused ontology and a benchmark of 30 MOP synthesis papers. The benchmark targets four information types: CBUs, characterisation entities, synthesis steps, and reaction chemicals. We did not test how results change when the ontology is updated and the pipeline is re-run. We also did not evaluate the approach in a different scientific domain.

Some extraction tasks remain harder than others. Chemical species identification shows high precision but lower recall, which means the system often misses valid mentions. This is likely due to variation in naming and reporting styles in the papers.

The main constraint on broader evaluation is the lack of curated datasets. Building reliable ground truth in new domains, or under revised ontologies, is still costly and time-consuming.

5.2 Future work

Future work will directly address the limitations identified above by systematically evaluating robustness under ontology change. We will modify the ontology and re-run the full compilation and instantiation pipeline to quantify how executable tool interfaces and prompts adapt to evolving schemas, and to what extent regeneration can replace manual pipeline re-engineering. In parallel, we will apply the framework to additional scientific domains with distinct ontological structures to assess transfer beyond MOP synthesis.

Where suitable curated data is available, we will expand evaluation to larger and more heterogeneous corpora and to additional extraction targets. Particular emphasis will be placed on recall-limited tasks, especially chemical species identification and normalisation, while preserving the current level of precision, in order to better characterise the trade-offs introduced by enforcing ontological constraints at generation time. These experiments will

allow us to quantify the stability of compiled action interfaces under schema and domain changes.

6 Methods

The methodology is organized as a staged pipeline that separates ontology-driven compilation, ontology-constrained KG construction, and grounding. The *preparation stage* compiles an ontology T-Box and a small set of domain-agnostic meta-prompts into executable artefacts. These include a static JSON iteration plan, ontology- and task-specific instantiation prompts, and an ontology-specific MCP server that exposes ontology-aware construction and validation tools. The *instantiation stage* executes the compiled plan for each document using a ReAct-style tool-use loop. It incrementally constructs A-Box individuals and relations in a persistent Turtle store and validates them as it goes. When constraints are violated, the tools return diagnostics that guide repair. The *grounding stage* aligns selected constructed instance IRIs to canonical entities in an existing knowledge graph. It generates an ontology-conditioned lookup interface from the target T-Box and endpoint evidence. It then applies deterministic lexical matching to produce an explicit IRI mapping. This mapping is applied by rewriting instance IRIs or by adding explicit `owl:sameAs` links. Domain-specific derivation modules (*e.g.* CBU derivation from crystallographic resources) are treated as downstream enrichment steps and are described separately from the core KG construction and grounding pipeline.

6.1 Preparation stage

The preparation stage converts an ontology T-Box into (i) a static, executable JSON plan for extraction and KG construction and (ii) an ontology-specific MCP server that exposes the operations referenced by that plan. The only manual inputs are a set of *domain-agnostic meta-prompts for extraction and KG construction* that are reused across papers and domains. These prompts are provided in Listings 7.2.1, 7.2.2, 7.2.3, and 7.2.4. As summarised in Figures 7.7 and 7.8, these meta-prompts guide an LLM to derive the task decomposition, synthesise ontology-aware scripts, materialise an MCP server, and generate instantiation prompts capturing task-specific interpretation rules.

JSON-based task decomposition. A *task decomposition agent* breaks the ontology-guided extraction problem into a small number of ordered steps. The result is written as a JSON plan, illustrated in Figure 7.8, which the instantiation agent can follow directly.

Each step in the plan states (i) what to do, (ii) what text prompt to use for extraction, and (iii) what prompt to use for knowledge-graph updates. The plan also lists what information each step reads and writes, such as the source paper, intermediate notes, and generated graph files. Optional sub-steps can be included for follow-up passes, such as enrichment or correction. Finally, the plan specifies which MCP tool groups are available at each step and which tools are needed, including their expected inputs and outputs. The outcome is a static, executable procedure that drives document processing in a fixed order.

Script and MCP tool generation from the T-Box. Given the T-Box and a set of domain-agnostic meta-prompts (Listings 7.2.1, 7.2.2, 7.2.3, and 7.2.4), the MCP Creation Agent generates an ontology-aware Python script that supports the classes and properties needed for the extraction scenario.

The script is built on top of a manually written, ontology-independent helper library. This library handles Turtle loading and saving, cross-step state management, deterministic IRI minting, and other generic utilities. The generated code is required to call these helpers rather than reimplement them. This keeps domain-specific logic separate from shared infrastructure.

The meta-prompts enforce a standard code structure. The script initialises an RDFLib graph, provides utilities to create and reuse instances by class, and defines functions to add links for each property. The generated code also distinguishes two kinds of constraints. *Hard* constraints come from the formal T-Box axioms. They include class hierarchy, domain and range typing, datatype restrictions, and any modelled cardinalities. These axioms determine tool inputs and outputs and drive run-time checks that prevent invalid triples. *Soft* constraints come from natural-language annotations in the ontology. For example, `rdfs:comment` may state inclusion or exclusion rules, naming conventions, or deduplication and reuse policies. These annotations are treated as guidance that complements, but does not override, the formal axioms.

MCP server construction. After the ontology-aware functions are generated, the MCP Integration Agent turns them into MCP tools. It reads each function signature and docstring and follows an integration meta-prompt. This process produces ontology-specific MCP servers, as shown in Figure 7.7.

The server exposes each function as an MCP tool with an ontology-derived name and a typed argument schema. It also attaches short usage instructions drawn from T-Box annotations. Tools are grouped into simple tool sets that match the JSON task plan, such as entity creation, attribute and relation completion, and cross-document linking. The resulting MCP server is then registered in the shared MCP tool pool and becomes available to all LLM-powered agents.

Design principles for instantiation MCP servers. Among the generated MCP servers, the instantiation MCP server is central at runtime, and its design principles are enforced during preparation. First, tool interfaces are defined in terms of ontology concepts, relations, and constraints: the meta-prompts require the underlying scripts to respect which classes may be connected by which properties, expected units, and (where applicable) reuse of concepts from reference ontologies (e.g. OM for units). Second, tool descriptions are ontology-guided: function signatures and natural-language instructions are derived from the T-Box, including `rdfs:comment` annotations that capture intended meanings and preferred usage patterns. Third, the server is wired to a persistent Turtle (`.ttl`) store that acts as cross-step memory, enabling reuse and incremental extension of previously created instances across multiple iterations.

Instantiation prompt generation In addition to tools and scripts, the preparation agent generates *domain- and task-specific instantiation prompts* that capture *soft interpretation constraints* not reliably encoded as formal axioms. These prompts encode operational definitions and heuristic decision rules that guide boundary setting and classification during extraction (*e.g.* what counts as a synthesis step and how to delimit it; how to recognise a `HeatChill` step; how to infer experimental atmosphere such as air vs. inert from textual cues). The instantiation agent uses these prompts alongside the JSON plan and the MCP tool schemas to decide what evidence to extract and when to invoke which tools; hard schema constraints are enforced by the ontology-aware tools during instance construction.

6.2 Instantiation stage

The instantiation stage executes the task specification produced during preparation to construct ontology-aligned knowledge graphs from documents. Guided by the JSON-based decomposition (Figure 7.8), the runtime follows the iteration-oriented structure shown in Figure 7.6, with an instantiation agent acting as a ReAct-style controller over MCP tools and agent-generated, runtime intermediate results (e.g. condensed passages and hints, extracted snippets, tool-call inputs/outputs, logs, and completion markers), together with a persistent Turtle store used for incremental KG updates.

Loading MCP tools for instantiation. Before processing a paper, the system loads two kinds of MCP servers. The first kind contains the ontology-specific instantiation tools generated in the preparation stage. The second kind provides shared utilities, such as external knowledge access and general text processing.

For chemical identifiers and properties, we use a third-party PubChem MCP server[1]. Access to crystallographic metadata is provided by a custom CCDC MCP server implemented in this work. A configuration file then lists the available tools and their input and output schemas. The instantiation agent reads this file so it can call both the local instantiation tools and the external PubChem and CCDC tools through a single, consistent interface.

Plan-driven ReAct execution. After the MCP tools are loaded, the instantiation agent follows the JSON plan step by step. Each step specifies the goal, the prompts to use, the files to read and write, and the tool groups that are allowed. Figure 7.8 shows an example plan step.

The agent runs each step in a ReAct-style loop. It first reads the paper and produces intermediate notes, such as condensed passages, extracted snippets, normalised names, and lookup results. It then calls ontology-specific MCP tools to create entities, add attributes, and link relations in the Turtle store. A step ends when the expected outputs for that step have been produced, including any follow-up passes used for enrichment.

[1] `PubChem-MCP-Server` (GitHub): https://github.com/JackKuo666/PubChem-MCP-Server

Each tool call returns both results and validation feedback. The feedback reports whether the requested update satisfies the ontology constraints. If a violation is detected, for example a missing required field, a type mismatch, or an invalid unit (Figure 1), the tool returns an error with an explanation. The agent then retries with corrected inputs. It may also extract missing evidence from the paper or query external resources before calling the tool again.

Ontology-constrained function calls. Ontology constraints are checked at tool-call time by the instantiation MCP server. The server functions follow the design rules set in the preparation stage. As illustrated in Figure 7.9, a function such as `create_temperature` looks up the relevant OM classes and properties. It checks the numeric value and unit against the T-Box. Only then does it create the corresponding individuals in the graph. Similar checks are applied to other quantities, relations, and class memberships.

Each call returns both the requested result and a check report. If the call violates a constraint, the server returns an error with an explanation. The instantiation agent uses this feedback in the next ReAct step to revise inputs, add missing fields, or trigger a repair action.

Turtle-based persistent memory. Throughout the instantiation stage, all MCP tools operate over a Turtle-encoded knowledge graph (`.ttl`) that serves as persistent cross-step memory. The instantiation MCP server reads from and writes to this store, updating it after each successful creation, enrichment, or repair operation. Instances created in earlier iterations can be looked up, linked, and incrementally extended in later ones, and identifiers are reused according to the deduplication and IRI-minting logic generated in the preparation scripts. Intermediate Turtle snapshots, together with the file-level `Inputs`/`Outputs` markers, record the state of the extraction run at each iteration, so that processing can be resumed from a given point or audited after the fact.

6.3 Grounding stage

After ontology-constrained KG construction, we apply a grounding stage to align constructed instance IRIs to canonical entities in a reference knowledge graph. The purpose is to normalize identity across documents and pipelines by linking locally minted IRIs to existing KG IRIs, enabling deduplication and integration. Grounding operates over selected ontology classes and produces a deterministic IRI mapping that is materialized either by rewriting IRIs or by adding `owl:sameAs` links.

Grounding tool generation Grounding tools are generated by LLM agents using the target ontology T-Box and KG endpoint evidence as the only domain-specific inputs. The goal of this stage is to compile a grounding interface that supports canonicalization of constructed instances against a reference knowledge graph.

Given a target ontology schema (T-Box) and a SPARQL endpoint for the existing knowledge graph, the grounding generation agents automatically synthesize three tool scripts.

First, a sampling script analyzes the ontology schema and endpoint to identify relevant classes and label-like predicates and to estimate instance distributions. Second, a label collection script collect and cache labels and identifiers for selected classes from the target KG and builds a local label index. Third, a query and lookup interface script is generated that provides SPARQL accessors and deterministic fuzzy-lookup functions over the local label index.

The resulting query and lookup interface is exposed as an ontology-specific MCP server and becomes available as callable grounding tools in the runtime pipeline. This generation process follows the same ontology-driven tool compilation paradigm as KG construction, while keeping grounding logic domain-agnostic at runtime.

Lexical grounding For each selected constructed instance, the grounding agent extracts one or more surface strings (*e.g.* `rdfs:label` and ontology-specific alternative name properties) and queries the target-KG MCP lookup server, which returns a ranked, finite set of candidate matches from a locally cached label index. The agent then applies a deterministic selection policy over this candidate set to choose a canonical target IRI and produces an explicit mapping from constructed IRIs to reference-KG IRIs. Finally, the mapping is materialized either by rewriting IRIs in the RDF graph or by adding `owl:sameAs` links; the same procedure supports both single-file and batch grounding over collections of Turtle outputs.

6.4 Preparation of evaluation dataset

We curated a benchmark of 30 scientific articles reporting MOP synthesis and characterisation, focusing on papers where the target entities and relations are explicitly described in the text. Ground truth was annotated in a predefined JSON record format inherited from the prior (non-MCP) workflow and retained here to enable like-for-like comparison across pipelines. This schema is aligned with the ontology T-Box in the sense that each task category specifies the corresponding entity types, relations, and attribute slots to be captured, but annotation is performed directly in JSON rather than as ontology instances.

We manually populated the JSON records for each paper following written guidelines that enforce consistent interpretation of entity/slot definitions and alignment with T-Box intent (*e.g.*, inclusion/exclusion criteria and expected value types). The resulting benchmark contains 6,705 annotated items across synthesis steps, reaction chemicals, characterisation entities, and chemical building units (CBUs) (Table 7.1).

6.5 Evaluation metrics and methodology

We evaluate end-to-end extraction *coupled with ontology instantiation* via a JSON-to-JSON protocol. For each task category, system outputs are first instantiated as ontology individuals and relations in a Turtle graph. We then apply a fixed, manually designed set of category-specific SPARQL queries (held constant across all runs) to retrieve the target individuals, relations, and literal attributes implied by the T-Box. Query bindings are

deterministically serialised into JSON records conforming to the same predefined schema used for annotation. This evaluates instantiation quality rather than text extraction alone: any missing individuals, missing links, or uninstantiated required slots are not returned by the SPARQL queries and are therefore counted as false negatives.

We compute precision, recall, and F1 by matching predicted and ground-truth JSON records using *slot-to-slot exact match* after basic normalisation. Normalisation is limited to non-semantic formatting fixes (*e.g.*, trimming whitespace, normalising casing where appropriate, and applying simple canonical forms for common unit strings when the schema expects a controlled label). For record sets where ordering is not semantically meaningful (notably synthesis steps), matching is *order-insensitive*: predicted records are aligned to ground truth under a one-to-one assignment that maximises the number of exactly matched slots, and any unmatched predicted or ground-truth records contribute to false positives or false negatives, respectively. Metrics are reported per category and aggregated over all papers; we additionally report micro-aggregated scores across all categories and macro-averaged scores across categories.

The same SPARQL-to-JSON conversion and matching procedure is applied to all comparative baselines and ablation settings, ensuring that all results reflect the same end-to-end criterion of producing evaluation-recoverable, correctly instantiated outputs. Finally, CBU items correspond to *grounded/derived* CBUs constructed by combining paper evidence with database-backed normalisation and derivation; they are therefore evaluated as an end-to-end grounding task rather than a pure surface-form extraction task.

6.6 CBU derivation for metal-organic polyhedra

In addition to ontology-driven extraction from textual synthesis procedures, the framework instantiates an agent for automated Chemical Building Unit (CBU) derivation for metal–organic polyhedra (MOPs). This agent is implemented as a ReAct-style controller that orchestrates MCP-based access to crystallographic and chemical databases, and then materialises the resulting CBUs directly into The World Avatar knowledge graph.

The CBU derivation workflow proceeds as follows:

- **MCP-based access to crystallographic data.** Given a MOP identifier (*e.g.* a CCDC deposition code), the agent uses an MCP server wrapping the CCDC database to locate and download the corresponding `.res` and `.cif` files. A Python-backed MCP tool parses these files to extract the asymmetric unit, symmetry operations, and atom/site information required for structural analysis.
- **Identification of CBUs.** Using the parsed crystallographic information, the agent invokes MCP tools for graph-based analysis of the coordination network (*e.g.* bond connectivity, coordination environments, linker topology). These tools segment the structure into metal nodes and organic linkers, and classify them into reusable CBU types suitable for downstream materials design workflows.
- **External chemical enrichment via PubChem and related services.** For each organic linker candidate, the agent calls MCP servers that wrap external databases

such as PubChem to retrieve standard identifiers (InChI, InChIKey, SMILES), synonyms, and basic physicochemical properties. This enrichment step normalises the CBUs against widely used chemical identifiers and supports cross-database linkage.

- **Ontology-aligned CBU instantiation.** The derived and enriched CBUs are then passed to the instantiation MCP server, which creates ontology-consistent instances representing metal nodes, linkers, and their connectivity. The resulting MOP CBU descriptions are written into the Turtle-based persistent memory and become part of the shared knowledge graph that can be reused by subsequent extraction and reasoning tasks.

By combining MCP-based access to CCDC, PubChem, and ontology-aware instantiation tools, the agent autonomously bridges crystallographic data and ontology-level CBU representations for MOPs, without manual scripting or ad hoc intermediate formats.

6.7 Models and inference settings

Components in the pipeline are driven by large language models (LLMs), with different models assigned to different roles. Unless otherwise stated, we use deterministic decoding for tool-calling (`temperature`=0) and enforce schema validation on structured outputs at tool boundaries. For non-tool free-form generations (*e.g.* narrative rationales or intermediate notes), we retain the same decoding settings unless explicitly noted to ensure reproducibility across runs.

Model assignments. We use `gpt-4.1` for document-level extraction; `gpt-4o` for knowledge-graph instantiation and construction actions; `gpt-5` for chemical building unit (CBU) derivation as well as script and prompt generation; and `gpt-4o-mini` for the agent-based query interface, where latency and cost are prioritised.

Inference and validation. Tool calls are executed with deterministic decoding (`temperature`=0) to minimise stochastic variation in argument selection and to make constraint-violation feedback comparable across runs. Structured outputs produced at tool boundaries are validated against the corresponding schemas, and violations are surfaced to the agent as explicit error messages to drive iterative repair until a valid instance is produced.

Token usage and cost. Script and prompt generation incurred a total LLM cost of $5.70. For end-to-end processing of 30 papers, the total LLM cost for document-level extraction and instantiation/grounding actions was $100.56 ($71.61 + $28.95), corresponding to an average per-article cost of $3.35.

Data availability

The curated benchmark dataset (30 MOP synthesis papers with >6,000 annotations), together with the curated ground truth, output files, and evaluation data used in this study, are available via the Cambridge Data Repository (doi:10.17863/CAM.126228) and via Dropbox[2].

Code availability

Code for the ontology-to-tools compilation workflow, including the MCP servers/tools and evaluation scripts, will be made available on GitHub[3].

Acknowledgements

This research was supported by the National Research Foundation, Prime Minister's Office, Singapore under its Campus for Research Excellence and Technological Enterprise (CREATE) programme. This project has received funding from the European Union's Horizon Europe research and innovation programme under grants 101074004 (C2IMPRESS), 101188248 (CLIMATE-ADAPT4EOSC), and 101226137 (TOGETHER).

M.K. gratefully acknowledges the support of the Alexander von Humboldt Foundation and the Massachusetts Institute of Technology. S.D.R. acknowledges financial support from Fitzwilliam College, Cambridge, and the Cambridge Trust.

We thank all contributors who assisted with data collection, annotation, and quality control for the manually curated MOP synthesis benchmark. In particular, we thank Mingxi Lu, Yichen Sun, Yuan Gao, Kuhan Thayalan, and Matthew Olatunji for annotation support.

Competing interests

The authors declare no competing interests.

[2]https://www.dropbox.com/scl/fi/u0dtyhyfa6cp7cr60jckq/Data-Public.zip
[3]https://github.com/TheWorldAvatar/mcp-tool-layer/

Author contributions

X.Z. contributed to ideation and conceptualisation, implemented the system, conducted the experiments, and wrote the manuscript. P.B. and C.Y. led data curation. T.A. contributed to data curation. J.A. contributed to ideation, conceptualisation, manuscript writing, and funding acquisition. S.R. contributed to data curation and provided domain expertise. S.M. contributed to data curation and manuscript writing. M.K. contributed to ideation, conceptualisation, manuscript writing, and funding acquisition. All authors reviewed and approved the final manuscript.

7 Supplementary Information

7.1 Supplementary Note 1: Background and related work

7.1.1 LLM-based agents and ReAct-style reasoning

Large language model (LLM)–based agents couple a generative model with an explicit action space, where each step conditions on a textual context (for example, a user request, intermediate results, and tool outputs) and selects the next action such as calling a tool, querying a knowledge graph, or producing a natural-language response. [54, 62, 65, 69] External capabilities are typically exposed through structured interfaces (for example, function signatures with typed arguments), and tool responses are returned as additional context to enable multi-step interaction with external systems. [54, 62]

ReAct-style agents organise this interaction as an explicit loop of *reasoning* and *acting*, alternating between intermediate reasoning traces and tool actions, and incorporating tool observations into subsequent decisions. [69] In information extraction, this framing supports decomposing long inputs, validating intermediate structures, and iteratively refining partial outputs through repeated reasoning–acting cycles.

7.1.2 Tool-calling interfaces and the Model Context Protocol

Earlier tool-calling interfaces for LLMs typically expose application-specific functions through proprietary schemas or prompt templates, and integration logic is often implemented separately for each deployment. [11, 32, 37] The Model Context Protocol (MCP) instead defines an open client–server protocol for connecting LLM applications to external tools and data sources. In MCP, servers register tools, resources, and prompts with machine-readable schemas; clients discover these capabilities and request tool invocations through a standardised messaging layer. [41, 44] MCP supports structured tool definitions, streaming of results, and standardised error reporting across heterogeneous hosts and models, enabling reuse of the same tool servers across different LLM applications.

7.1.3 The World Avatar and its chemistry ontologies

The World Avatar (TWA) is a cross-domain digital ecosystem that represents entities, processes, and their interdependencies using ontologies and RDF knowledge graphs, accessed and modified by software agents at runtime. [5] The reticular-chemistry stack of TWA is organised around two domain-generic core ontologies and one application ontology. OntoSpecies represents chemical species, identifiers, and characterisation data (with extensions for IR and elemental analysis). OntoSyn (OntoSynthesis) represents synthesis procedures as sequences of `SynthesisStep` operations transforming input `Species` into `ChemicalOutput` products. OntoMOPs acts as an application ontology for metal–organic polyhedra (MOPs), modelling MOP instances and their building units and linking OntoSyn procedures and OntoSpecies reactants into the MOP design space. [5, 29, 46, 51] Previous work implemented a MOP synthesis pipeline in TWA that used LLMs guided by

ontology-aligned JSON schemas to populate these chemistry knowledge graphs. [51]

7.1.4 LLM-based knowledge instantiation and ontology grounding

Large language models are increasingly used to construct and maintain knowledge graphs, by inducing schemas, populating instances, or interacting with graph backends. [33, 50] From the perspective of this work, two lines are most relevant: schema-level approaches that define or refine extraction targets, and instance-level approaches that extract and validate concrete assertions.

Schema-level (T-Box). Schema-level pipelines treat the extraction schema as a contract that defines what types, fields, and constraints outputs should satisfy. PARSE refines JSON Schemas as first-class artefacts for extraction, but optimises primarily syntactic constraints encoded at the JSON-schema level rather than ontology axioms. [57] LLMs4SchemaDiscovery mines candidate scientific schemas from text with human feedback, again focusing on schema discovery rather than using a fixed ontology T-Box as the contract. [53] SCHEMA-MINERpro adds ontology grounding by mapping discovered schemas to reference ontologies, but grounding remains a stage separate from instance creation. [52] AutoSchemaKG induces schemas and triples at scale without a predefined domain T-Box, enabling cross-domain graphs but decoupling extraction from ontology-encoded constraints such as units and identification rules. [4]

Instance-level (A-Box). Instance-level pipelines focus on extracting entities, relations, and events as concrete assertions. KGGen uses strict JSON specifications to extract and refine triples, improving structural consistency, but semantic validity with respect to a concrete ontology is enforced via post-hoc checks rather than constraint-aware instance construction. [39] Other pipelines align extracted relations to ontology predicates or use ontologies for filtering and validation, but typically keep ontology enforcement separate from the tool interfaces that govern how instances are created. [27, 28, 42]

MCP-assisted LLM-based solutions. MCP has emerged as a general mechanism for connecting LLM applications to tools and data sources. [41, 44] Existing MCP-based knowledge-graph servers often expose generic CRUD-style interfaces over graph backends, demonstrating tool-based access but typically leaving ontological correctness to application logic or post-hoc validation rather than compiling T-Box axioms into tool signatures and runtime checks. [21]

TWA agent composition framework Prior to recent LLM-based agent systems, The World Avatar (TWA) explored how semantic descriptions can support the *discovery*, *composition*, and *execution* of software agents within a knowledge-graph ecosystem. [80] introduced a semantic agent composition framework for TWA, in which agents are described using a lightweight agent ontology and grounded execution metadata to enable automated composition of agent workflows across domains.

This work motivates a view of ontologies not only as data models but also as interfaces to execution, where semantic descriptions govern which agents can be composed and how they may interact. In contrast to agent discovery and composition based on pre-defined interfaces, the present work compiles domain ontologies into executable tool interfaces and run-time constraints that directly regulate generative behaviour.

7.2 Supplementary Methods

This section provides extended implementation and evaluation details referenced in the main Methods, including dataset curation and annotation protocol, evaluation metrics and matching rules (including the CBU formula-only criterion), MCP server/tool specifications, and runtime policies (iterations, error handling, and repair strategies).

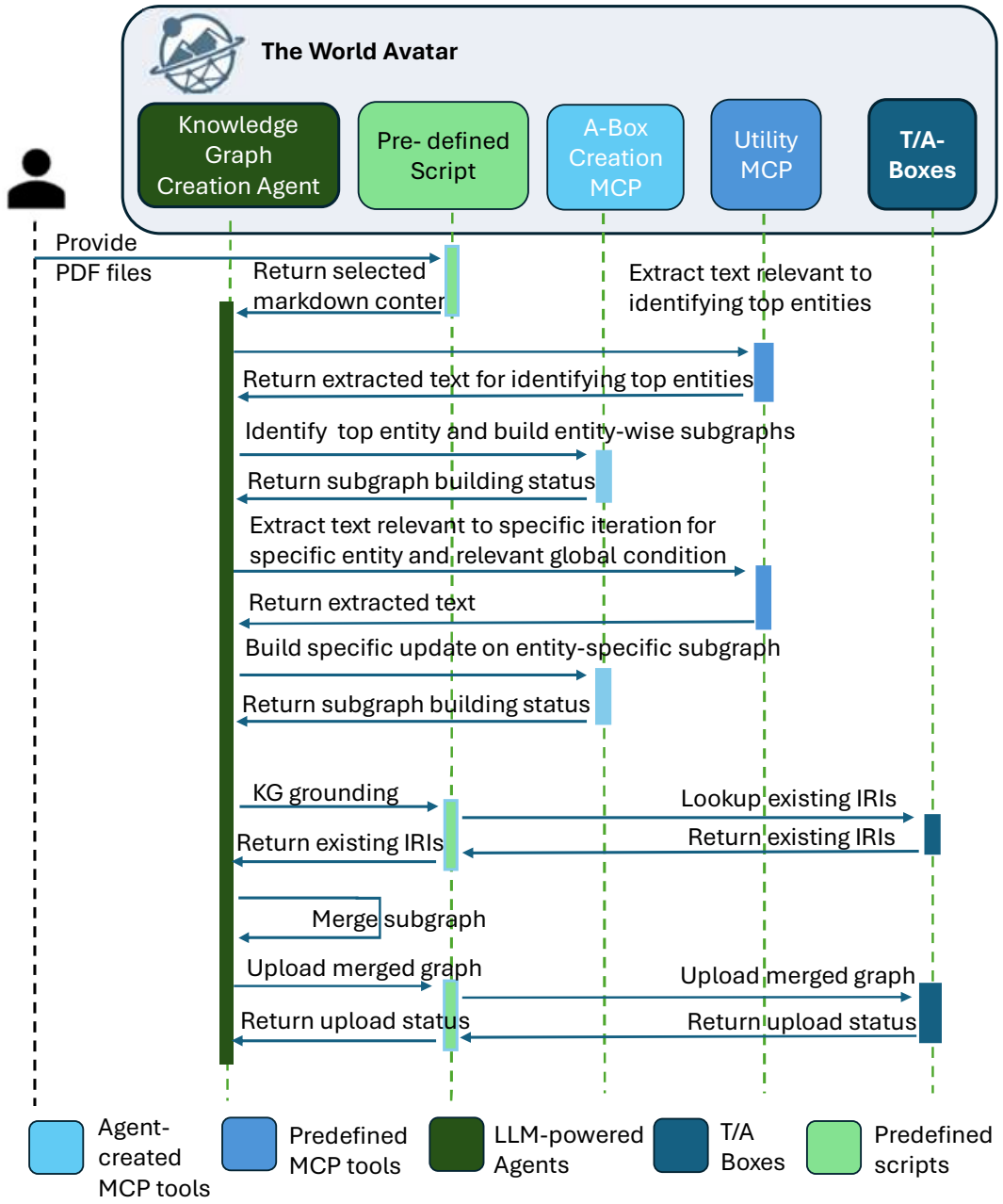

Figure 7.6: *Ontology-to-tools compilation workflow. A preparation agent takes as input the ontology T-Box and a small set of manually authored, domain-agnostic meta-prompts for extraction and KG construction. It synthesises (i) an ontology-specific MCP server exposing ontology-aware tools with machine-checkable schemas and (ii) domain- and task-specific instantiation prompts (for synthesis steps, reaction chemicals, characterisation entities, and CBUs) that steer the runtime agent. A ReAct-style instantiation agent then executes the generated prompts and invokes the MCP tools to construct knowledge-graph instances from documents.*

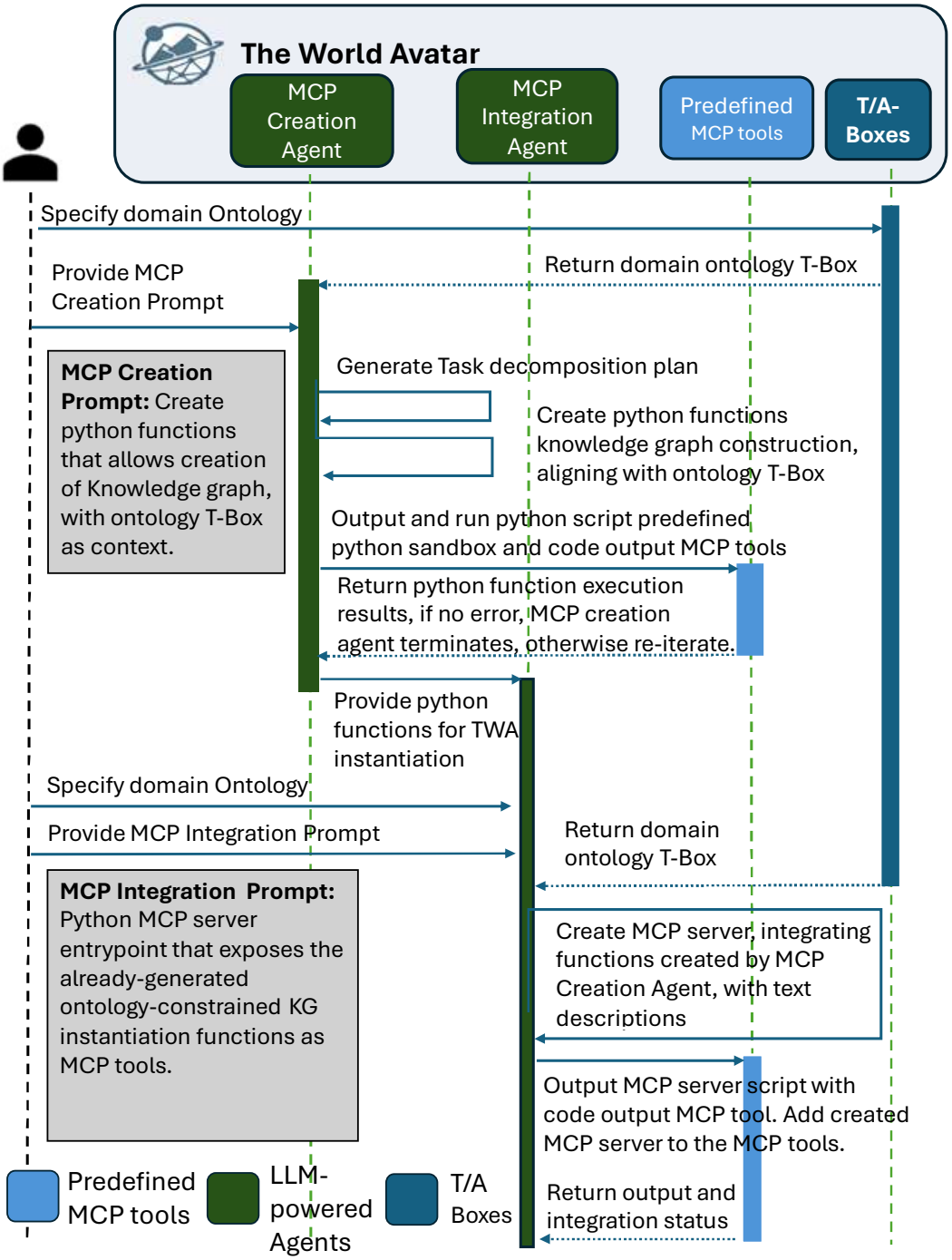

Figure 7.7: *UML workflow of the two-agent framework for ontology-aligned knowledge-graph construction within the World Avatar (TWA) environment. The **MCP Creation Agent** runs in the TWA context and takes the domain ontology (T-Box; schema) and an MCP Creation Prompt to generate a KG instantiation library. The **MCP Integration Agent** then takes an MCP Integration Prompt plus the validated library (and its function descriptions) to package it into a deployable MCP server, exposing these functions as callable MCP tools and registering them into a shared MCP tool pool for reuse by downstream LLM agents, including TWA workflows.*

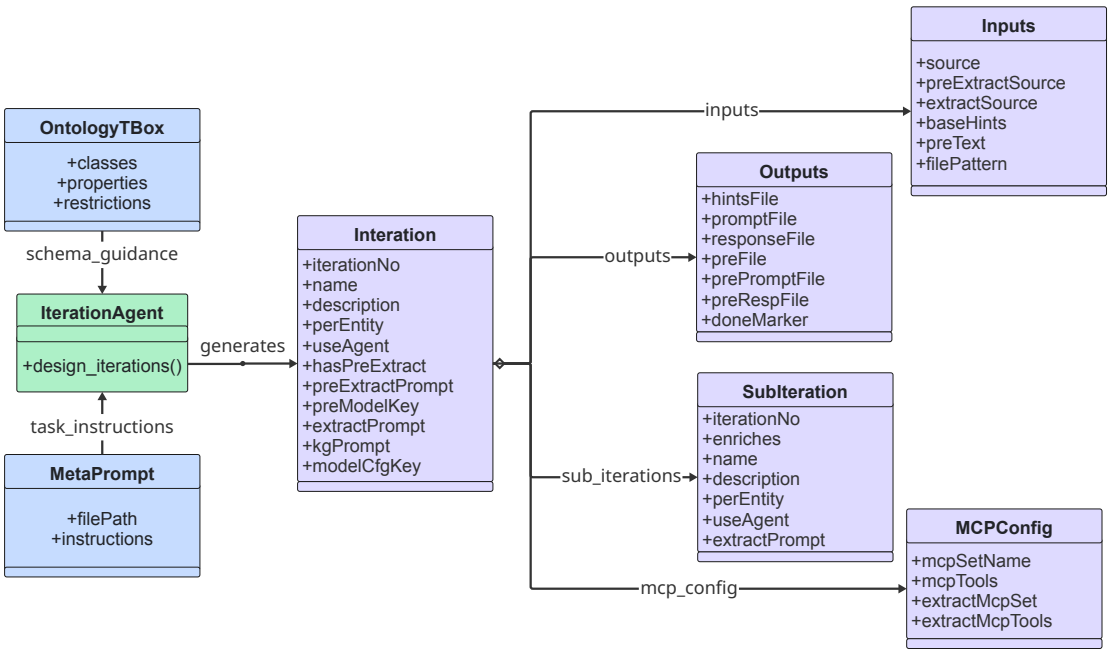

Figure 7.8: *Class diagram of the structured task specification: a task decomposition model, guided by the Ontology T-Box and a meta-prompt, generates iterations that bundle and KG-building steps, file flows (inputs/outputs), optional sub-iterations, and MCP tool configurations.*

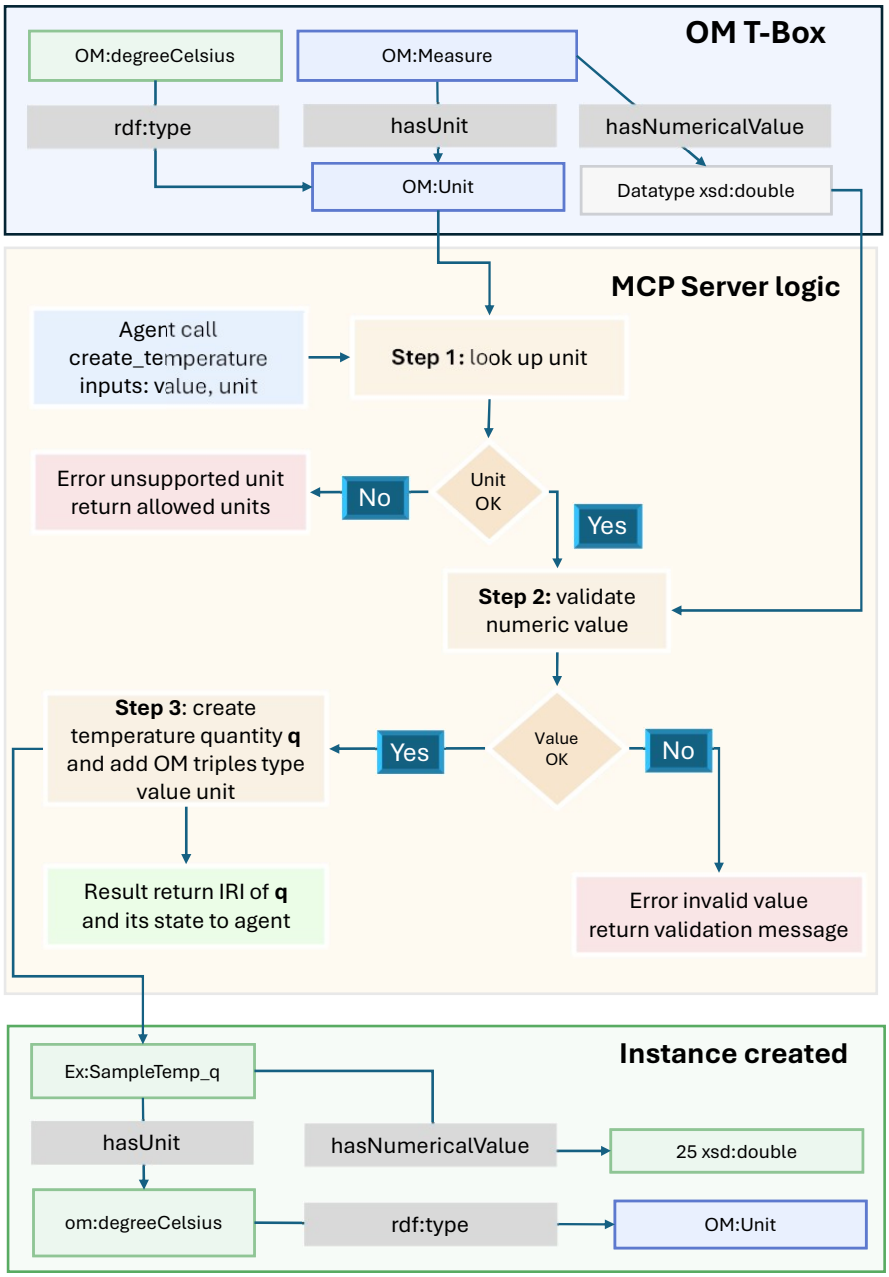

Figure 7.9: *Illustrative example of the interaction between the OM T-Box (top), a single* `create_temperature` *call within the MCP server logic (middle), and the resulting OM instance (bottom). The top box shows the ontology T-Box, which defines schema-level constraints for temperature quantities, such as the relevant classes and properties. The middle box shows an example script call that enforces these constraints by looking up the unit, validating the numeric value, and only then creating the corresponding OM individuals for the temperature value and its unit. The bottom box shows the resulting example instance that satisfies the T-Box constraints, linking the created quantity to a valid unit and a compliant numerical value.*

7.2.1 Iteration 1: System Meta Prompt

```
You are a knowledge graph construction prompt expert specializing in
    ITERATION 1 KG building prompts.

ITERATION 1 is special: it only creates top-level entity instances from
    paper content, WITHOUT creating any related entities (inputs,
    outputs, sub-components, or detailed steps).

Your task:
1. Analyze the provided ontology (T-Box) to identify the top entity
    type (the main class that represents the overall procedure/process)
2. Extract all relevant rules, constraints, and identification
    guidelines from rdfs:comment annotations
3. Generate a focused, tool-oriented KG building prompt for ITERATION 1

CRITICAL CONSTRAINTS FOR ITER1:
- ONLY create instances of the top-level entity class (one per
    procedure described in the paper)
- Do NOT create any related entities (inputs, outputs, sub-components,
    steps) in this iteration
- Link each top-level entity to its source document
- Apply strict identification rules from the ontology to determine what
    qualifies as a valid instance
- Include all cardinality, scope, exclusion, and linking rules from the
    ontology

OUTPUT REQUIREMENTS:
- Start with "Follow these generic rules for any iteration." followed
    by the global rules
- Include MCP tool-specific guidance (error handling, IRI management,
    check_existing_* usage)
- Include explicit identification section (document identifier handling)
- Clearly state the task scope: create top-level entities only, no
    related entities
- List all constraints from the ontology rdfs:comment for the top-level
    entity class
- Include clear termination conditions
- Be concise and tool-oriented (this prompt is for an MCP agent using
    function calls)
- Be completely domain-agnostic (no specific compound types, no
    domain-specific terminology)

Do NOT include:
- Domain-specific examples (\eg specific compound names, specific
    synthesis types)
- Variable placeholders like {doi} or {paper_content} - these will be
    added programmatically
```

```
- Verbose ontology explanations - focus on actionable rules
- Any mention of specific chemical entities, materials, or
    domain-specific concepts
```

7.2.2 Iteration 1: User Meta Prompt

```
Based on the ontology and available MCP tools below, generate a KG
    building prompt for ITERATION 1.

ITERATION 1 SCOPE:
- Create ONLY top-level entity instances (the main procedure/process
    class)
- Do NOT create any related entities (inputs, outputs, sub-components,
    steps)
- Link to source document

ONTOLOGY (T-Box):
{tbox}

MCP MAIN SCRIPT (Available Tools) [Python]:
{mcp_main_script}

The MCP Main Script shows all available tools with their descriptions,
    parameters, and usage guidance. Use this to understand what tools
    are available and how they should be called.

REQUIREMENTS:
1. Extract ALL rules from the top-level entity class rdfs:comment,
    especially:
   - Scope (what qualifies as a valid instance)
   - Different forms / methods / variations rules
   - Exclusions (what NOT to create)
   - Cardinality requirements
   - Linking requirements
   - Conservative behavior guidelines
   - Critical exclusions for extraction

2. Include global MCP rules:
   - Tool invocation rules (never call same tool twice with identical
       args)
   - IRI management (must create before passing, use check_existing_*
       tools)
   - Error handling (status codes, already_attached, retryable)
   - Placeholder policies
   - Termination conditions (run_status: done)
```

```
3. Include identification section:
   - Document identifier handling (treat as sole task identifier, reuse
     consistently)
   - Entity focus guidance (for when entity_label/entity_uri provided)

4. Be concise and actionable - this is for an MCP agent making function
   calls

5. Be completely domain-agnostic:
   - Do NOT mention specific compound types, materials, or chemical
     entities
   - Use generic terminology that applies to any domain
   - Adapt wording from the ontology to be domain-neutral where possible

Generate the prompt now (do NOT include variable placeholders - those
   will be added programmatically):
```

7.2.3 Extension Ontology: System Meta Prompt

```
You are an expert in creating knowledge graph building prompts for
   extension ontologies.

Extension ontologies are simpler ontologies that extend a main ontology
   with additional specialized information. They use MCP tools to
   build A-Boxes that link to the main ontology's A-Box.

Your task is to analyze a T-Box ontology and MCP tools to create a KG
   building prompt that:
1. Provides a clear task route for building the extension A-Box
2. Emphasizes using MCP tools to populate the A-Box
3. Requires comprehensive population (making certain MCP function calls
   compulsory)
4. Emphasizes IRI reuse from the main ontology A-Box
5. Includes domain-specific requirements extracted from the T-Box
   comments
6. Forbids fabrication - only use information from the paper

CRITICAL RULES:
- Read ALL classes, properties, and rdfs:comment fields in the T-Box
- Extract domain-specific requirements from rdfs:comment (\eg required
   fields, cardinality constraints)
- Focus on HOW to build the A-Box, not just WHAT to extract
- Emphasize the connection between the extension A-Box and the main
   ontology A-Box
- Make the prompt actionable with clear steps
- Output ONLY the prompt text (no markdown fences, no commentary)
```

7.2.4 Extension Ontology: User Meta Prompt

```
Generate a KG building prompt for an extension ontology.

T-Box (analyze to understand the ontology structure and requirements):
```turtle
{tbox}
MCP Main Script (understand available tools, their parameters, and
   calling sequences):

python

{mcp_main_script}
Your prompt MUST:

State the task - Extend the main ontology A-Box with the extension A-Box

Provide a task route - Give a recommended sequence of steps for
   building the KG

Emphasize MCP tools - Instruct to use the MCP server to populate the
   A-Box

Require IRI reuse - Emphasize reusing existing IRIs from the main
   ontology A-Box

Extract T-Box requirements - Include any compulsory requirements from
   rdfs:comment (\eg required fields, minimum instances)

Forbid fabrication - Only use information from the paper content

Include tool-specific notes - If the T-Box or domain requires special
   handling (\eg external database integration, data transformations),
   include those notes

Structure your output as:

css

Your task is to extend the provided A-Box of [MainOntology] with the
   [extension ontology] A-Box, according to the paper content.

You should use the provided MCP server to populate the [extension
   ontology] A-Box.

Here is the recommended route of task:

[Step-by-step guidance based on T-Box structure]
```
```

```
Requirements:

[List of requirements based on T-Box rdfs:comment and MCP tool
   constraints]

Special note:

[Any domain-specific notes based on T-Box comments]

Here is the DOI for this run (normalized and pipeline forms):

- DOI: {{doi_slash}}
- Pipeline DOI: {{doi_underscore}}

Here is the [MainOntology] A-Box:

{{main_ontology_a_box}}

Here is the paper content:

{{paper_content}}

CRITICAL:

Extract domain-specific requirements from the T-Box rdfs:comment
   fields. Do NOT invent requirements. ALL requirements must be
   justified by the T-Box or MCP tool constraints.

Output EXACTLY the structure shown above. Do NOT add any additional
   sections after {{paper_content}}. This is the END of the prompt.

Generate the prompt now:
```

7.2.5 Illustrative trace of constraint-triggered repair

Listing 1 shows a representative repair cycle triggered by an ontology constraint on measurement units. The ontology restricts temperature units to a controlled vocabulary that includes *degree Celsius*. When the instantiation agent first instantiates a temperature condition using the verbatim unit token *C* from text, the ontology-aware MCP tool rejects the input and returns a structured validation error listing allowed values. The agent then normalises the unit to an allowed entry and retries, producing a valid instance.

Listing 1: *Representative MCP trace: ontology-constrained unit repair (C → degree Celsius).*

```
Input evidence (paper text):
  "... heated at 120 C for 12 h ..."

Attempt 1 (verbatim unit token):
  Tool: mop.create_temperature_condition
  Input:    { "context": "reaction_step_3",
              "value": 120,
              "unit": "C" }

  Tool response (constraint violation):
    { "ok": false,
      "error_type": "OntologyConstraintViolation",
      "field": "unit",
      "message": "Unit value 'C' is not permitted by the ontology.",
      "allowed_values": ["degree Celsius", "kelvin"]
    }

Repair (from tool feedback):
  - Map shorthand unit token "C" to the allowed vocabulary entry
     "degree Celsius".
  - Retry tool call with corrected unit.

Attempt 2 (normalised unit):
  Tool: mop.create_temperature_condition
  Input:
    { "context": "reaction_step_3",
      "value": 120,
      "unit": "degree Celsius"
    }

  Tool response (success):
    {
      "ok": true,
      "instance_iri": "twa:TemperatureCondition_0f3a...",
      "validated": true
    }
```

7.3 Supplementary Evaluation Results

Tables 7.4–7.6 report per-paper extraction performance against the full ground truth for the three evaluated outputs: synthesis steps (Table 7.4), chemical building units under a formula-only matching criterion (Table 7.5), and characterisation entities (Table 7.6). For each paper (indexed by DOI), we provide true positives (TP), false positives (FP), false negatives (FN), and the derived precision, recall, and F1 scores, followed by an overall aggregate row.

Table 7.1: *Benchmark profile (30 papers). Ground-truth positives are computed as TP+FN from Table 7.2.*

Category	Papers	Ground-truth positives
CBU	30	164
Characterisation	30	926
Steps	30	4940
Chemicals	30	675
Total	30	6705

7.3.1 Detailed instance visualisation for a UMC-1 synthesis

The Supplementary Information includes a detailed visualisation of one instantiated ontology subgraph corresponding to the synthesis of the metal–organic polyhedron UMC-1 (Supplementary Fig. 7.10). The figure shows the complete set of instantiated individuals and relations for this synthesis recipe under the OntoSyn and OntoMOPs ontologies, including typed synthesis steps, chemical inputs, and the resulting product entity, without the abstraction and condensation used in the main-text example (Fig. 1).

The graph was initially rendered from the RDF serialisation using an RDF-to-Graphviz visualisation tool[4] and then manually edited to adjust the layout for presentation and page fitting.

[4] https://giacomociti.github.io/rdf2dot/

Table 7.2: *Overall evaluation summary by category (system output recovered from instantiated graphs vs. full ground truth). CBU denotes grounded/derived CBUs (Methods).*

Category	TP	FP	FN	Precision	Recall	F1
CBU	128	40	36	0.762	0.780	0.771
Characterisation	669	102	257	0.868	0.722	0.788
Steps	4225	857	715	0.831	0.855	0.843
Chemicals	393	0	282	1.000	0.582	0.736
Micro-aggregate	**5415**	**999**	**1290**	**0.844**	**0.808**	**0.826**
Macro-average	—	—	—	**0.865**	**0.735**	**0.785**

Table 7.3: *Ablation study. CBU denotes grounded/derived CBUs (Methods).*

Setting	CBU F1	Char. F1	Steps F1	Chem. F1
Full system	**0.771**	**0.788**	**0.843**	**0.736**
– external tools	0.000	0.610	0.800	0.732
– constraint feedback	0.768	0.788	0.572	0.724

Table 7.4: *Per-paper extraction results for synthesis steps against the full ground truth.*

DOI	TP	FP	FN	Precision	Recall	F1
10.1002.anie.201811027 [14]	185	31	19	0.856	0.907	0.881
10.1002.anie.202010824 [15]	305	10	19	0.968	0.941	0.955
10.1002.chem.201604264 [35]	245	28	30	0.897	0.891	0.894
10.1002.chem.201700798 [25]	47	10	8	0.825	0.855	0.839
10.1002.chem.201700848 [13]	111	20	30	0.847	0.787	0.816
10.1021.acs.cgd.6b00306 [49]	35	10	4	0.778	0.897	0.833
10.1021.acs.chemmater.8b01667 [7]	105	35	24	0.750	0.814	0.781
10.1021.acs.inorgchem.4c02394 [63]	216	38	44	0.850	0.831	0.840
10.1021.acs.inorgchem.8b01130 [17]	207	74	70	0.737	0.747	0.742
10.1021.acsami.7b09339 [45]	117	35	40	0.770	0.745	0.757
10.1021.acsami.7b18836 [31]	80	24	6	0.769	0.930	0.842
10.1021.acsami.8b02015 [6]	118	28	37	0.808	0.761	0.784
10.1021.cg4018322 [47]	50	4	10	0.926	0.833	0.877
10.1021.ic050460z [26]	139	63	41	0.688	0.772	0.728
10.1021.ic402428m [34]	180	51	36	0.779	0.833	0.805
10.1021.ic501012e [59]	147	7	3	0.955	0.980	0.967
10.1021.ic802382p [48]	84	14	5	0.857	0.944	0.898
10.1021.ja042802q [58]	389	135	80	0.742	0.829	0.783
10.1021.ja105986b [77]	63	23	25	0.733	0.716	0.724
10.1021.jacs.7b00037 [72]	128	4	7	0.970	0.948	0.959
10.1021.jacs.8b10866 [73]	214	51	28	0.808	0.884	0.844
10.1039.C2CC34265K [12]	166	22	33	0.883	0.834	0.858
10.1039.C5CC05913E [2]	55	13	4	0.809	0.932	0.866
10.1039.C5DT04764A [76]	161	0	3	1.000	0.982	0.991
10.1039.C5RA26357C [8]	50	14	14	0.781	0.781	0.781
10.1039.C6CC04583A [75]	206	64	52	0.763	0.798	0.780
10.1039.C6DT02764D [74]	103	15	15	0.873	0.873	0.873
10.1039.C7CC01208J [71]	92	10	10	0.902	0.902	0.902
10.1039.C8DT02580K [16]	119	5	8	0.960	0.937	0.948
10.1039.D3QI01501G [68]	106	20	12	0.841	0.898	0.869
Overall	4223	858	717	0.831	0.855	0.843

Table 7.5: *Per-paper formula-only extraction results for chemical building units (CBUs) against the full ground truth.*

DOI	TP	FP	FN	Precision	Recall	F1
10.1021.acsami.7b18836 [31]	3	1	1	0.750	0.750	0.750
10.1039.C8DT02580K [16]	4	0	0	1.000	1.000	1.000
10.1002.chem.201700848 [13]	4	2	2	0.667	0.667	0.667
10.1021.acs.inorgchem.4c02394 [63]	8	2	2	0.800	0.800	0.800
10.1021.ic402428m [34]	8	0	0	1.000	1.000	1.000
10.1039.C5RA26357C [8]	2	0	0	1.000	1.000	1.000
10.1039.C6DT02764D [74]	2	2	2	0.500	0.500	0.500
10.1021.ic501012e [59]	4	0	0	1.000	1.000	1.000
10.1002.chem.201604264 [35]	8	2	2	0.800	0.800	0.800
10.1021.jacs.7b00037 [72]	1	3	3	0.250	0.250	0.250
10.1039.C7CC01208J [71]	2	2	2	0.500	0.500	0.500
10.1021.acschemmater.8b01667 [7]	6	2	0	0.750	1.000	0.857
10.1039.C5DT04764A [76]	3	3	3	0.500	0.500	0.500
10.1039.C6CC04583A [75]	10	0	0	1.000	1.000	1.000
10.1021.ic802382p [48]	2	0	0	1.000	1.000	1.000
10.1002.anie.202010824 [15]	7	3	3	0.700	0.700	0.700
10.1021.acs.inorgchem.8b01130 [17]	4	2	2	0.667	0.667	0.667
10.1039.C2CC34265K [12]	3	3	3	0.500	0.500	0.500
10.1021.acsami.7b09339 [45]	8	0	0	1.000	1.000	1.000
10.1002.anie.201811027 [14]	4	4	0	0.500	1.000	0.667
10.1021.acs.cgd.6b00306 [49]	0	2	2	0.000	0.000	0.000
10.1039.D3QI01501G [68]	2	0	2	1.000	0.500	0.667
10.1021.cg4018322 [47]	2	0	0	1.000	1.000	1.000
10.1039.C5CC05913E [2]	2	0	0	1.000	1.000	1.000
10.1021.ja105986b [77]	2	0	0	1.000	1.000	1.000
10.1021.ic050460z [26]	6	0	0	1.000	1.000	1.000
10.1021.jacs.8b10866 [73]	6	2	2	0.750	0.750	0.750
10.1021.ja042802q [58]	7	5	3	0.583	0.700	0.636
10.1021.acsami.8b02015 [6]	6	0	2	1.000	0.750	0.857
10.1002.chem.201700798 [25]	2	0	0	1.000	1.000	1.000
Overall	128	40	36	0.762	0.780	0.771

Table 7.6: *Per-paper extraction results for characterisation entities against the full ground truth.*

DOI	TP	FP	FN	Precision	Recall	F1
10.1002.anie.201811027 [14]	26	0	4	1.000	0.867	0.929
10.1002.anie.202010824 [15]	36	11	16	0.766	0.692	0.727
10.1002.chem.201604264 [35]	10	7	16	0.588	0.385	0.465
10.1002.chem.201700798 [25]	5	2	5	0.714	0.500	0.588
10.1002.chem.201700848 [13]	21	4	15	0.840	0.583	0.689
10.1021.acs.cgd.6b00306 [49]	8	1	1	0.889	0.889	0.889
10.1021.acs.chemmater.8b01667 [7]	36	3	10	0.923	0.783	0.847
10.1021.acs.inorgchem.4c02394 [63]	43	9	16	0.827	0.729	0.775
10.1021.acs.inorgchem.8b01130 [17]	38	1	11	0.974	0.776	0.864
10.1021.acsami.7b09339 [45]	29	1	11	0.967	0.725	0.829
10.1021.acsami.7b18836 [31]	17	2	2	0.895	0.895	0.895
10.1021.acsami.8b02015 [6]	19	2	8	0.905	0.704	0.792
10.1021.cg4018322 [47]	13	1	17	0.929	0.433	0.591
10.1021.ic050460z [26]	31	1	7	0.969	0.816	0.886
10.1021.ic402428m [34]	20	0	24	1.000	0.455	0.625
10.1021.ic501012e [59]	14	4	6	0.778	0.700	0.737
10.1021.ic802382p [48]	7	2	5	0.778	0.583	0.667
10.1021.ja042802q [58]	51	2	6	0.962	0.895	0.927
10.1021.ja105986b [77]	8	2	5	0.800	0.615	0.696
10.1021.jacs.7b00037 [72]	16	6	4	0.727	0.800	0.762
10.1021.jacs.8b10866 [73]	41	8	9	0.837	0.820	0.828
10.1039.C2CC34265K [12]	23	8	13	0.742	0.639	0.687
10.1039.C5CC05913E [2]	7	2	4	0.778	0.636	0.700
10.1039.C5DT04764A [76]	24	2	6	0.923	0.800	0.857
10.1039.C5RA26357C [8]	8	2	5	0.800	0.615	0.696
10.1039.C6CC04583A [75]	43	10	15	0.811	0.741	0.775
10.1039.C6DT02764D [74]	18	2	4	0.900	0.818	0.857
10.1039.C7CC01208J [71]	18	2	4	0.900	0.818	0.857
10.1039.C8DT02580K [16]	16	4	4	0.800	0.800	0.800
10.1039.D3QI01501G [68]	22	4	11	0.846	0.667	0.746
Overall	668	105	264	0.864	0.717	0.784

Table 7.7: *Per-paper extraction results for chemicals against the full ground truth.*

DOI	TP	FP	FN	Precision	Recall	F1
10.1002.anie.201811027 [14]	8	0	6	1.000	0.571	0.727
10.1002.anie.202010824 [15]	24	0	6	1.000	0.800	0.889
10.1002.chem.201604264 [35]	22	0	13	1.000	0.629	0.772
10.1002.chem.201700798 [25]	3	0	6	1.000	0.333	0.500
10.1002.chem.201700848 [13]	12	0	7	1.000	0.632	0.774
10.1021.acs.cgd.6b00306 [49]	5	0	3	1.000	0.625	0.769
10.1021.acs.chemmater.8b01667 [7]	7	0	1	1.000	0.875	0.933
10.1021.acs.inorgchem.4c02394 [63]	24	0	14	1.000	0.632	0.774
10.1021.acs.inorgchem.8b01130 [17]	7	0	25	1.000	0.219	0.359
10.1021.acsami.7b09339 [45]	5	0	1	1.000	0.833	0.909
10.1021.acsami.7b18836 [31]	11	0	3	1.000	0.786	0.880
10.1021.acsami.8b02015 [6]	7	0	4	1.000	0.636	0.778
10.1021.cg4018322 [47]	5	0	14	1.000	0.263	0.417
10.1021.ic050460z [26]	12	0	3	1.000	0.800	0.889
10.1021.ic402428m [34]	28	0	5	1.000	0.848	0.918
10.1021.ic501012e [59]	18	0	12	1.000	0.600	0.750
10.1021.ic802382p [48]	7	0	7	1.000	0.500	0.667
10.1021.ja042802q [58]	31	0	14	1.000	0.689	0.816
10.1021.ja105986b [77]	6	0	5	1.000	0.545	0.706
10.1021.jacs.7b00037 [72]	14	0	13	1.000	0.519	0.683
10.1021.jacs.8b10866 [73]	30	0	29	1.000	0.508	0.674
10.1039.C2CC34265K [12]	17	0	15	1.000	0.531	0.694
10.1039.C5CC05913E [2]	6	0	6	1.000	0.500	0.667
10.1039.C5DT04764A [76]	12	0	26	1.000	0.316	0.480
10.1039.C5RA26357C [8]	7	0	9	1.000	0.438	0.609
10.1039.C6CC04583A [75]	23	0	12	1.000	0.657	0.793
10.1039.C6DT02764D [74]	11	0	5	1.000	0.688	0.815
10.1039.C7CC01208J [71]	11	0	9	1.000	0.550	0.710
10.1039.C8DT02580K [16]	11	0	3	1.000	0.786	0.880
10.1039.D3QI01501G [68]	9	0	6	1.000	0.600	0.750
Overall	393	0	282	1.000	0.582	0.736

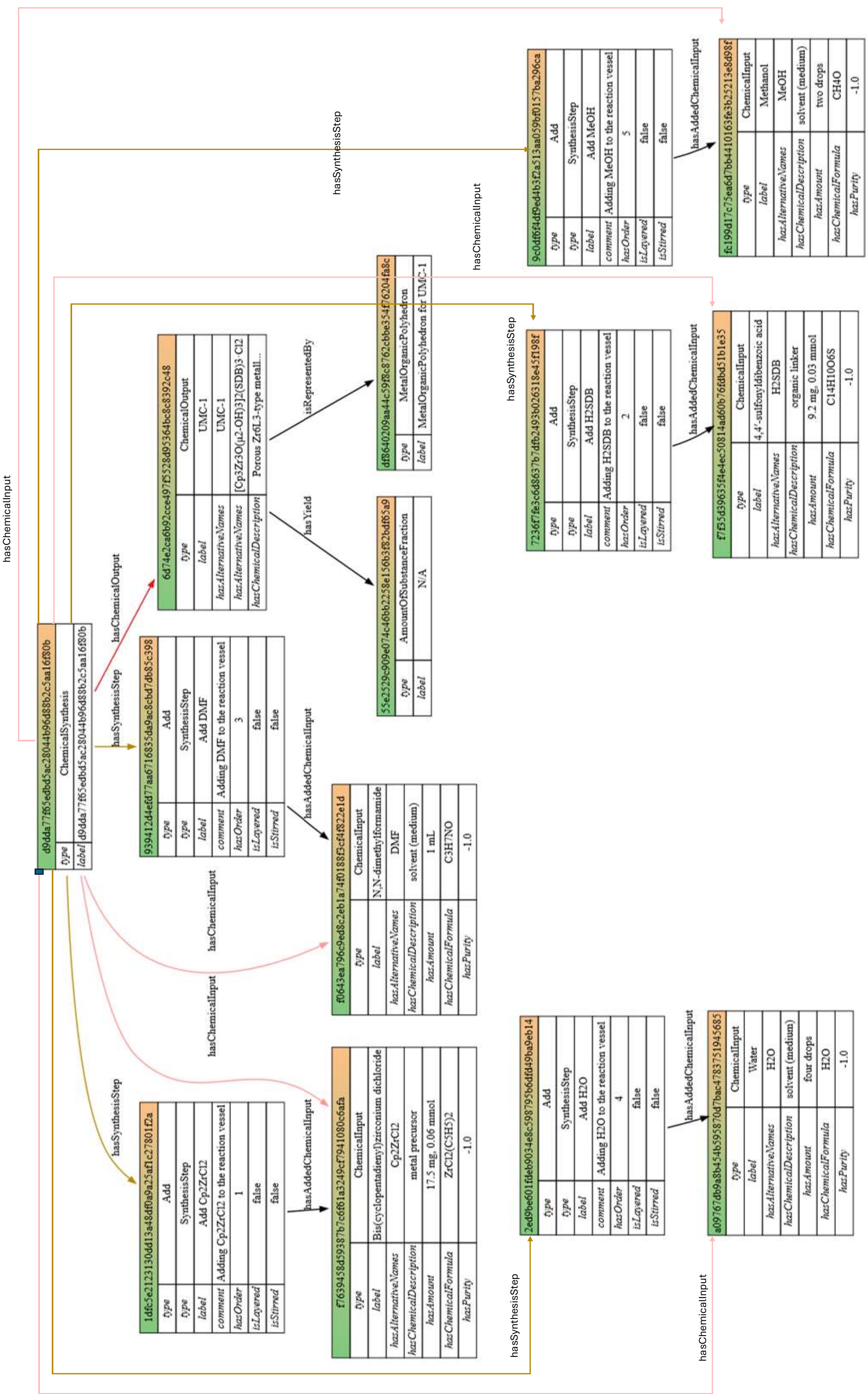

Figure 7.10: ***Detailed ontology instance subgraph for a UMC-1 synthesis.***

References

[1] Anthropic. Introducing the model context protocol. https://www.anthropic.com/news/model-context-protocol, Nov. 2024. Accessed: 2025-12-18.

[2] A. W. Augustyniak, M. Fandzloch, M. Domingo, I. Łakomska, and J. A. R. Navarro. A vanadium(iv) pyrazolate metal–organic polyhedron with permanent porosity and adsorption selectivity. *Chemical Communications*, 51(79):14724–14727, 2015. doi:10.1039/C5CC05913E.

[3] J. Babutzka, M. Bortz, A. Dinges, G. Foltin, D. Hajnal, H. Schultze, and H. Weiss. Machine Learning Supporting Experimental Design for Product Development in the Lab. *Chemie Ingenieur Technik*, 91(3):277–284, Mar. 2019. ISSN 0009-286X, 1522-2640. doi:10.1002/cite.201800089. URL https://onlinelibrary.wiley.com/doi/10.1002/cite.201800089.

[4] J. Bai, W. Fan, Q. Hu, Q. Zong, C. Li, H. T. Tsang, H. Luo, Y. Yim, H. Huang, X. Zhou, F. Qin, T. Zheng, X. Peng, X. Yao, H. Yang, L. Wu, Y. Ji, G. Zhang, R. Chen, and Y. Song. Autoschemakg: Autonomous knowledge graph construction through dynamic schema induction from web-scale corpora. *arXiv preprint arXiv:2505.23628*, 2025. doi:10.48550/arXiv.2505.23628.

[5] J. Bai, S. D. Rihm, A. Kondinski, F. Saluz, X. Deng, G. Brownbridge, S. Mosbach, J. Akroyd, and M. Kraft. twa: The world avatar python package for dynamic knowledge graphs and its application in reticular chemistry. *Digital Discovery*, 4: 2123–2135, 2025. doi:10.1039/D5DD00069F.

[6] O. Barreda, G. Bannwart, G. P. A. Yap, and E. D. Bloch. Ligand-Based Phase Control in Porous Molecular Assemblies. *ACS Applied Materials & Interfaces*, 10 (14):11420–11424, Apr. 2018. doi:10.1021/acsami.8b02015.

[7] O. Barreda, G. A. Taggart, C. A. Rowland, G. P. A. Yap, and E. D. Bloch. Mechanochemical Synthesis of Porous Molecular Assemblies. *Chemistry of Materials*, 30(12):3975–3978, June 2018. doi:10.1021/acs.chemmater.8b01667.

[8] Z. Chen, X. Liu, A. Wu, Y. Liang, X. Wang, and F. Liang. Synthesis, structure and properties of an octahedral dinuclear-based Cu_{12} nanocage of trimesoyltri(l -alanine). *RSC Advances*, 6(12):9911–9915, 2016. doi:10.1039/C5RA26357C.

[9] C. W. Coley, R. Barzilay, T. S. Jaakkola, W. H. Green, and K. F. Jensen. Prediction of Organic Reaction Outcomes Using Machine Learning. *ACS Central Science*, 3(5):434–443, May 2017. ISSN 2374-7943, 2374-7951. doi:10.1021/acscentsci.7b00064. URL https://pubs.acs.org/doi/10.1021/acscentsci.7b00064.

[10] C. W. Coley, D. A. Thomas, J. A. M. Lummiss, J. N. Jaworski, C. P. Breen, V. Schultz, T. Hart, J. S. Fishman, L. Rogers, H. Gao, R. W. Hicklin, P. P. Plehiers, J. Byington, J. S. Piotti, W. H. Green, A. J. Hart, T. F. Jamison, and K. F.

Jensen. A robotic platform for flow synthesis of organic compounds informed by AI planning. *Science*, 365(6453):eaax1566, Aug. 2019. ISSN 0036-8075, 1095-9203. doi:10.1126/science.aax1566. URL https://www.science.org/doi/10.1126/science.aax1566.

[11] T. Dave, S. A. Athaluri, and S. Singh. ChatGPT in medicine: an overview of its applications, advantages, limitations, future prospects, and ethical considerations. *Frontiers in Artificial Intelligence*, 6:1169595, May 2023. ISSN 2624-8212. doi:10.3389/frai.2023.1169595. URL https://www.frontiersin.org/articles/10.3389/frai.2023.1169595/full.

[12] S. Du, C. Hu, J.-C. Xiao, H. Tan, and W. Liao. A giant coordination cage based on sulfonylcalix[4]arenes. *Chemical Communications*, 48(73):9177, 2012. doi:10.1039/c2cc34265k.

[13] B. Garai, A. Mallick, A. Das, R. Mukherjee, and R. Banerjee. Self-Exfoliated Metal-Organic Nanosheets through Hydrolytic Unfolding of Metal-Organic Polyhedra. *Chemistry – A European Journal*, 23(30):7361–7366, May 2017. doi:10.1002/chem.201700848.

[14] Y. Gong, Y. Zhang, C. Qin, C. Sun, X. Wang, and Z. Su. Bottom-Up Construction and Reversible Structural Transformation of Supramolecular Isomers based on Large Truncated Tetrahedra. *Angewandte Chemie International Edition*, 58(3):780–784, Jan. 2019. doi:10.1002/anie.201811027.

[15] Y. Gong, C. Qin, Y. Zhang, C. Sun, Q. Pan, X. Wang, and Z. Su. Face-Directed Assembly of Molecular Cubes: In Situ Substitution of a Predetermined Concave Cluster. *Angewandte Chemie International Edition*, 59(49):22034–22038, Dec. 2020. doi:10.1002/anie.202010824.

[16] Y.-R. Gong, W.-C. Chen, L. Zhao, K.-Z. Shao, X.-L. Wang, and Z.-M. Su. Functionalized polyoxometalate-based metal–organic cuboctahedra for selective adsorption toward cationic dyes in aqueous solution. *Dalton Transactions*, 47(37):12979–12983, 2018. doi:10.1039/C8DT02580K.

[17] A. J. Gosselin, C. A. Rowland, K. P. Balto, G. P. A. Yap, and E. D. Bloch. Design and Synthesis of Porous Nickel(II) and Cobalt(II) Cages. *Inorganic Chemistry*, 57 (19):11847–11850, Oct. 2018. doi:10.1021/acs.inorgchem.8b01130.

[18] R. Grishman. Information extraction. *IEEE Intelligent Systems*, 30(5):8–15, 2015. doi:10.1109/MIS.2015.68.

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

[20] N. Guarino, D. Oberle, and S. Staab. What is an ontology? In S. Staab and R. Studer, editors, *Handbook on Ontologies*, pages 1–17. Springer, Berlin, Heidelberg, 2 edition, 2009. doi:10.1007/978-3-540-92673-3_0.

[21] G. Hall. Memento MCP: A knowledge graph memory system for LLMs. https://github.com/gannonh/memento-mcp, 2025. Accessed: 2025-12-09.

[22] A. Hogan, E. Blomqvist, M. Cochez, C. D'amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, A.-C. N. Ngomo, A. Polleres, S. M. Rashid, A. Rula, L. Schmelzeisen, J. Sequeda, S. Staab, and A. Zimmermann. Knowledge Graphs. *ACM Computing Surveys*, 54(4):1–37, May 2022. ISSN 0360-0300, 1557-7341. doi:10.1145/3447772. URL https://dl.acm.org/doi/10.1145/3447772.

[23] K. M. Jablonka, P. Schwaller, A. Ortega-Guerrero, and B. Smit. Leveraging large language models for predictive chemistry. *Nature Machine Intelligence*, 6(2):161–169, 2024. doi:10.1038/s42256-023-00788-1.

[24] Y. Jiang, Y. Yu, M. Kong, Y. Mei, L. Yuan, Z. Huang, K. Kuang, Z. Wang, H. Yao, J. Zou, C. W. Coley, and Y. Wei. Artificial Intelligence for Retrosynthesis Prediction. *Engineering*, 25:32–50, June 2023. ISSN 20958099. doi:10.1016/j.eng.2022.04.021. URL https://linkinghub.elsevier.com/retrieve/pii/S2095809922005665.

[25] Z. Ju, G. Liu, Y. Chen, D. Yuan, and B. Chen. From Coordination Cages to a Stable Crystalline Porous Hydrogen-Bonded Framework. *Chemistry – A European Journal*, 23(20):4774–4777, Apr. 2017. doi:10.1002/chem.201700798.

[26] Y. Ke, D. J. Collins, and H.-C. Zhou. Synthesis and Structure of Cuboctahedral and Anticuboctahedral Cages Containing 12 Quadruply Bonded Dimolybdenum Units. *Inorganic Chemistry*, 44(12):4154–4156, June 2005. doi:10.1021/ic050460z.

[27] S. Khorshidi, A. Nikfarjam, S. Shankar, Y. Sang, Y. Govind, H. Jang, A. Kasgari, A. McClimans, M. Soliman, V. Konda, A. Fakhry, and X. Qi. Odke+: Ontology-guided open-domain knowledge extraction with LLMs. *arXiv preprint arXiv:2509.04696*, 2025. doi:10.48550/arXiv.2509.04696.

[28] V. K. Kommineni, B. König-Ries, and S. Samuel. From human experts to machines: An LLM supported approach to ontology and knowledge graph construction. *arXiv preprint arXiv:2403.08345*, 2024. doi:10.48550/arXiv.2403.08345.

[29] A. Kondinski, A. Menon, D. Nurkowski, F. Farazi, S. Mosbach, J. Akroyd, and M. Kraft. Automated rational design of metal–organic polyhedra. *Journal of the American Chemical Society*, 144(26):11713–11728, 2022. doi:10.1021/jacs.2c03402.

[30] M. Krallinger, O. Rabal, A. Lourenço, J. Oyarzabal, and A. Valencia. Information retrieval and text mining technologies for chemistry. *Chemical Reviews*, 117(12):7673–7761, 2017. doi:10.1021/acs.chemrev.6b00851.

[31] S. Lee, J. H. Lee, J. C. Kim, S. Lee, S. K. Kwak, and W. Choe. Porous Zr_6 L_3 Metallocage with Synergetic Binding Centers for CO_2. *ACS Applied Materials & Interfaces*, 10(10):8685–8691, Mar. 2018. doi:10.1021/acsami.7b18836.

[32] D. Li, Y. Zhao, Z. Wang, C. Jung, and Z. Zhang. Large Language Model-Driven Structured Output: A Comprehensive Benchmark and Spatial Data Generation Framework. *ISPRS International Journal of Geo-Information*, 13(11):405, Nov. 2024. ISSN 2220-9964. doi:10.3390/ijgi13110405. URL https://www.mdpi.com/2220-9964/13/11/405.

[33] X. Liang, Z. Wang, M. Li, and Z. Yan. A survey of LLM-augmented knowledge graph construction and application in complex product design. *Procedia CIRP*, 128: 870–875, 2024. doi:10.1016/j.procir.2024.07.069.

[34] G. Liu, Z. Ju, D. Yuan, and M. Hong. In Situ Construction of a Coordination Zirconocene Tetrahedron. *Inorganic Chemistry*, 52(24):13815–13817, Dec. 2013. doi:10.1021/ic402428m.

[35] G. Liu, M. Zeller, K. Su, J. Pang, Z. Ju, D. Yuan, and M. Hong. Controlled Orthogonal Self-Assembly of Heterometal-Decorated Coordination Cages. *Chemistry – A European Journal*, 22(48):17345–17350, Nov. 2016. doi:10.1002/chem.201604264.

[36] J. Liu, M. Zhang, W. Li, C. Wang, S. Li, H. Jiang, S. Jiang, Y. Xiao, and Y. Chen. Beyond Entities: A Large-Scale Multi-Modal Knowledge Graph with Triplet Fact Grounding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):18653–18661, Mar. 2024. ISSN 2374-3468, 2159-5399. doi:10.1609/aaai.v38i17.29828. URL https://ojs.aaai.org/index.php/AAAI/article/view/29828.

[37] M. X. Liu, F. Liu, A. J. Fiannaca, T. Koo, L. Dixon, M. Terry, and C. J. Cai. "We Need Structured Output": Towards User-centered Constraints on Large Language Model Output. In *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, pages 1–9, Honolulu HI USA, May 2024. ACM. ISBN 979-8-4007-0331-7. doi:10.1145/3613905.3650756. URL https://dl.acm.org/doi/10.1145/3613905.3650756.

[38] S. J. Martínez-Rodríguez, A. Hogan, and A. Meroño-Peñuela. Ontology-based information extraction for the semantic web: A survey. *Semantic Web*, 11(2):255–335, 2020. doi:10.3233/SW-180333.

[39] B. Mo, K. Yu, J. Kazdan, P. Mpala, L. Yu, C. Cundy, C. Kanatsoulis, and S. Koyejo. Kggen: Extracting knowledge graphs from plain text with language models. *arXiv preprint arXiv:2502.09956*, 2025. doi:10.48550/arXiv.2502.09956.

[40] Model Context Protocol Project. Model context protocol specification (version 2025-11-25). https://modelcontextprotocol.io/specification/2025-11-25, Nov. 2025. Accessed: 2025-12-18.

[41] Model Context Protocol Project. Model Context Protocol specification. https://modelcontextprotocol.io/specification/, 2025. Accessed: 2025-12-04.

[42] O. R. Olasunkanmi, M. Satursky, H. Yi, C. Bizon, H. Lee, and S. Ahalt. Relate: Relation extraction in biomedical abstracts with LLMs and ontology constraints. *arXiv preprint arXiv:2509.19057*, 2025. doi:10.48550/arXiv.2509.19057.

[43] OpenAI. Function calling and structured output formatting in the openai api. https://platform.openai.com/docs/guides/structured-outputs, 2024.

[44] OpenAI. Model Context Protocol in the OpenAI Agents SDK. https://platform.openai.com/docs/mcp, 2025. Accessed: 2025-12-04.

[45] J. Park, Z. Perry, Y.-P. Chen, J. Bae, and H.-C. Zhou. Chromium(II) Metal–Organic Polyhedra as Highly Porous Materials. *ACS Applied Materials & Interfaces*, 9(33): 28064–28068, Aug. 2017. doi:10.1021/acsami.7b09339.

[46] L. Pascazio, S. D. Rihm, A. Naseri, S. Mosbach, J. Akroyd, and M. Kraft. Chemical species ontology for data integration and knowledge discovery. *Journal of Chemical Information and Modeling*, 63(21):6569–6586, 2023. doi:10.1021/acs.jcim.3c00820.

[47] M. Paul, N. N. Adarsh, and P. Dastidar. Secondary Building Unit (SBU) Controlled Formation of a Catalytically Active Metal-Organic Polyhedron (MOP) Derived from a Flexible Tripodal Ligand. *Crystal Growth & Design*, 14(3):1331–1337, Mar. 2014. doi:10.1021/cg4018322.

[48] M. J. Prakash, Y. Zou, S. Hong, M. Park, M.-P. N. Bui, G. H. Seong, and M. S. Lah. Metal–organic polyhedron based on a Cu^{ii} paddle-wheel secondary building unit at the truncated octahedron corners. *Inorganic Chemistry*, 48(4):1281–1283, Feb. 2009. doi:10.1021/ic802382p.

[49] A. S. Rathnayake, K. A. Feaster, J. White, C. L. Barnes, S. J. Teat, and J. L. Atwood. Investigating Reaction Conditions To Control the Self-Assembly of Cobalt-Seamed Nanocapsules. *Crystal Growth & Design*, 16(7):3562–3564, July 2016. doi:10.1021/acs.cgd.6b00306.

[50] X. Ren, J. Tang, D. Yin, N. Chawla, and C. Huang. A survey of large language models for graphs. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6616–6626, 2024. doi:10.1145/3637528.3671460.

[51] S. D. Rihm, F. Saluz, A. Kondinski, J. Bai, P. W. V. Butler, S. Mosbach, J. Akroyd, and M. Kraft. Extraction of chemical synthesis information using the world avatar. *Digital Discovery*, 4(10):2893–2909, 2025. doi:10.1039/D5DD00183H.

[52] S. Sadruddin, J. D'Souza, E. Poupaki, A. Watkins, B. Karasulu, S. Auer, A. Mackus, and E. Kessels. Schema-minerpro: Agentic AI for ontology grounding over LLM-discovered scientific schemas in a human-in-the-loop workflow. *Semantic Web Journal*, 2025. In review.

[53] S. Sadruddin, J. D'Souza, E. Poupaki, A. Watkins, B. Karasulu, S. Auer, A. Rula, H. B. Giglou, A. Mackus, and E. Kessels. Llms4schemadiscovery: A human-in-the-loop workflow for scientific schema mining with large language models. In *The Semantic Web. ESWC 2025*, pages 244–261, 2025. doi:10.1007/978-3-031-94578-6_14.

[54] T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom. Toolformer: Language Models Can Teach Themselves to Use Tools, 2023. URL https://arxiv.org/abs/2302.04761. Version Number: 1.

[55] M. Schilling-Wilhelmi, M. Ríos-García, S. Shabih, M. V. Gil, S. Miret, C. T. Koch, J. A. Márquez, and K. M. Jablonka. From text to insight: large language models for chemical data extraction. *Chemical Society Reviews*, 54(3):1125–1150, 2025. ISSN 0306-0012, 1460-4744. doi:10.1039/D4CS00913D. URL https://xlink.rsc.org/?DOI=D4CS00913D.

[56] P. Schwaller, A. C. Vaucher, R. Laplaza, C. Bunne, A. Krause, C. Corminboeuf, and T. Laino. Machine intelligence for chemical reaction space. *WIREs Computational Molecular Science*, 12(5):e1604, Sept. 2022. ISSN 1759-0876, 1759-0884. doi:10.1002/wcms.1604. URL https://wires.onlinelibrary.wiley.com/doi/10.1002/wcms.1604.

[57] A. Shrimal, A. Jain, S. Chowdhury, and P. Yenigalla. Parse: LLM driven schema optimization for reliable entity extraction. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 2749–2763, 2025. doi:10.18653/v1/2025.emnlp-industry.184.

[58] A. C. Sudik, A. R. Millward, N. W. Ockwig, A. P. Côté, J. Kim, and O. M. Yaghi. Design, Synthesis, Structure, and Gas (N_2 , Ar, CO_2 , CH_4 , and H_2) Sorption Properties of Porous Metal-Organic Tetrahedral and Heterocuboidal Polyhedra. *Journal of the American Chemical Society*, 127(19):7110–7118, May 2005. doi:10.1021/ja042802q.

[59] H. Tan, S. Du, Y. Bi, and W. Liao. Two Elongated Octahedral Coordination Cages Constructed by M_4 -TC4A Secondary Building Units (M = Co^{ii} and Fe^{ii}) and 2,2'-Bipyridine-4,4'-dicarboxylic Acids. *Inorganic Chemistry*, 53(14):7083–7085, July 2014. doi:10.1021/ic501012e.

[60] Y. Tew, L. Fu, J. Li, T. Sun, L. Li, and X. Wang. Construction of Knowledge Graph for Material and Energy Flow Integration in Biochemical Industry, Jan. 2024. URL https://www.energy-proceedings.org/?p=11062.

[61] Z. Tu, T. Stuyver, and C. W. Coley. Predictive chemistry: machine learning for reaction deployment, reaction development, and reaction discovery. *Chemical Science*, 14(2):226–244, 2023. ISSN 2041-6520, 2041-6539. doi:10.1039/D2SC05089G. URL https://xlink.rsc.org/?DOI=D2SC05089G.

[62] L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei, and J. Wen. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345, Dec. 2024. ISSN 2095-2228, 2095-2236. doi:10.1007/s11704-024-40231-1. URL https://link.springer.com/10.1007/s11704-024-40231-1.

[63] Y.-H. Wang, K.-W. Tong, S.-R. Xiong, C.-Q. Chen, Y.-H. Song, and P. Yang. Steerable Structural Evolvement and Adsorption Behavior of Metastable

Polyoxovanadate-Based Metal–Organic Polyhedra. *Inorganic Chemistry*, 63(44): 20984–20992, Nov. 2024. doi:10.1021/acs.inorgchem.4c02394.

[64] D. C. Wimalasuriya and D. Dou. Ontology-based information extraction: An introduction and survey of current approaches. *Journal of Information Science*, 36(3): 306–323, 2010. doi:10.1177/0165551509360123.

[65] Q. Wu, G. Bansal, J. Zhang, Y. Wu, B. Li, E. Zhu, L. Jiang, X. Zhang, S. Zhang, J. Liu, A. H. Awadallah, R. W. White, D. Burger, and C. Wang. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation, 2023. URL https://arxiv.org/abs/2308.08155. Version Number: 2.

[66] D. Xu, W. Chen, W. Peng, C. Zhang, T. Xu, X. Zhao, X. Wu, Y. Zheng, Y. Wang, and E. Chen. Large language models for generative information extraction: a survey. *Frontiers of Computer Science*, 18(6):186357, Dec. 2024. ISSN 2095-2228, 2095-2236. doi:10.1007/s11704-024-40555-y. URL https://link.springer.com/10.1007/s11704-024-40555-y.

[67] W. Xu et al. Constrained decoding methods for structured text generation with large language models. *arXiv preprint arXiv:2302.XXXX*, 2023. To be updated with full bibliographic details once confirmed.

[68] Y. Yang, Y. Fu, Y. Tian, L. Zhao, C. Qin, X. Wang, and Z. Su. Efficient iodine capture by metal–organic cubes based on hexanuclear vanadium clusters. *Inorganic Chemistry Frontiers*, 10(21):6221–6228, 2023. doi:10.1039/D3QI01501G.

[69] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. doi:10.48550/arXiv.2210.03629. URL https://arxiv.org/abs/2210.03629.

[70] H. Ye, N. Zhang, H. Chen, and H. Chen. Generative Knowledge Graph Construction: A Review. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1–17, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics. doi:10.18653/v1/2022.emnlp-main.1. URL https://aclanthology.org/2022.emnlp-main.1.

[71] C. Zhang, R. S. Patil, T. Li, C. L. Barnes, and J. L. Atwood. Self-assembly of magnesium-seamed hexameric pyrogallol[4]arene nanocapsules. *Chemical Communications*, 53(31):4312–4314, 2017. doi:10.1039/C7CC01208J.

[72] C. Zhang, R. S. Patil, C. Liu, C. L. Barnes, and J. L. Atwood. Controlled 2D Assembly of Nickel-Seamed Hexameric Pyrogallol[4]arene Nanocapsules. *Journal of the American Chemical Society*, 139(8):2920–2923, Mar. 2017. doi:10.1021/jacs.7b00037.

[73] Y. Zhang, H. Gan, C. Qin, X. Wang, Z. Su, and M. J. Zaworotko. Self-Assembly of Goldberg Polyhedra from a Concave $[WV_5\ O_{11}\ (RCO_2\ )_5\ (SO_4\ )]^{3-}$ Building Block with 5-Fold Symmetry. *Journal of the American Chemical Society*, 140(50):17365–17368, Dec. 2018. doi:10.1021/jacs.8b10866.

[74] Y.-T. Zhang, S.-B. Li, X.-L. Wang, Y.-R. Gong, K.-Z. Shao, and Z.-M. Su. Synthesis, structures, and magnetic properties of metal–organic polyhedra based on unprecedented $\{V_7\}$ isopolyoxometalate clusters. *Dalton Transactions*, 45(38): 14898–14901, 2016. doi:10.1039/C6DT02764D.

[75] Y.-T. Zhang, X.-L. Wang, S.-B. Li, Y.-R. Gong, B.-Q. Song, K.-Z. Shao, and Z.-M. Su. Anderson-like alkoxo-polyoxovanadate clusters serving as unprecedented second building units to construct metal–organic polyhedra. *Chemical Communications*, 52(62):9632–9635, 2016. doi:10.1039/C6CC04583A.

[76] Y.-T. Zhang, X.-L. Wang, E.-L. Zhou, X.-S. Wu, B.-Q. Song, K.-Z. Shao, and Z.-M. Su. Polyoxovanadate-based organic–inorganic hybrids: from $\{V_5\ O_9\ Cl\}$ clusters to nanosized octahedral cages. *Dalton Transactions*, 45(9):3698–3701, 2016. doi:10.1039/C5DT04764A.

[77] S.-T. Zheng, J. Zhang, X.-X. Li, W.-H. Fang, and G.-Y. Yang. Cubic Polyoxometalate-Organic Molecular Cage. *Journal of the American Chemical Society*, 132(43):15102–15103, Nov. 2010. doi:10.1021/ja105986b.

[78] T. Zheng, Z. Deng, H. T. Tsang, W. Wang, J. Bai, Z. Wang, and Y. Song. From Automation to Autonomy: A Survey on Large Language Models in Scientific Discovery. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 17744–17761, Suzhou, China, 2025. Association for Computational Linguistics. doi:10.18653/v1/2025.emnlp-main.895. URL https://aclanthology.org/2025.emnlp-main.895.

[79] Z. Zheng, O. Zhang, H. L. Nguyen, N. Rampal, A. H. Alawadhi, Z. Rong, T. Head-Gordon, C. Borgs, J. T. Chayes, and O. M. Yaghi. Chatgpt research group for optimizing the crystallinity of MOFs and COFs. *ACS Central Science*, 9(11):2161–2170, 2023. doi:10.1021/acscentsci.3c01087.

[80] X. Zhou, A. Eibeck, M. Q. Lim, N. B. Krdzavac, and M. Kraft. An agent composition framework for the j-park simulator – a knowledge graph for the process industry. *Computers & Chemical Engineering*, 130:106577, 2019. doi:10.1016/j.compchemeng.2019.106577.