



Cite this: DOI: 10.1039/d6dd00301j

# A software framework for translating literature-derived $\chi$ DL procedures into execution-format programs on the Chemspeed automation platform

Yuuya Nagata,<sup>a</sup> Kojiro Machi,<sup>b</sup> Seiji Akiyama<sup>c</sup> and Masaharu Yoshioka<sup>bde</sup>

`xd12chemspeed` is a software framework that translates literature-derived  $\chi$ DL procedures into execution-format programs for the Chemspeed automation platform. The workflow reads  $\chi$ DL files, uses an LLM-assisted mapping stage to generate an intermediate Chemspeed command CSV, and applies a deterministic downstream converter for CSV quoting, zone renaming, unit normalisation, reflux-temperature substitution, global loading scaling, and reagent-position table generation. A deterministic XML-parsing reference converter was implemented to test the  $\chi$ DL-to-CSV mapping independently of the LLM. For four *Organic Syntheses* procedures containing 47  $\chi$ DL actions, the reference converter and LLM-assisted route produced byte-for-byte identical intermediate and final CSV files after downstream conversion. The reference route also writes row-level provenance sidecars that link each generated row to the source  $\chi$ DL step, source parameter, normalized value, conversion rule, software version, and file hashes, while preserving the Chemspeed execution CSV schema. An expert fidelity audit assigned 44 actions as correctly expanded and 3 as intentionally omitted because the current Chemspeed execution format has no corresponding field, with no action-level mistranslations. Ten independent runs with remote and local language models gave byte-for-byte identical final execution files after deterministic post-processing. The generated program for the benzylation of 3,4-dimethoxyphenol was executed on a Chemspeed SWING workstation without manual CSV correction, affording 4-benzyloxy-1,2-dimethoxybenzene in 84.4% isolated yield at 2.65 mmol scale. The framework provides an auditable bridge between structured synthesis procedures and commercial laboratory automation hardware.

Received 22nd May 2026  
Accepted 14th August 2026

DOI: 10.1039/d6dd00301j

rsc.li/digitaldiscovery

## 1 Introduction

The digitization of synthetic chemistry is increasingly important because organic synthesis procedures are still communicated mainly as prose in journal articles and patents. Such prose is informative for trained chemists, but it is intrinsically difficult to search, normalize, validate, and execute on automated platforms. This limitation is particularly acute for materials science, where discovery and optimization of functional materials such as photocatalysts, organic semiconductors, and battery electrolytes depend on rapid screening of molecular and formulation spaces. Recent reviews have established self-

driving laboratories and materials acceleration platforms as a research paradigm in which automated experimentation, data infrastructure, and algorithmic decision-making are integrated into a closed experimental loop.<sup>1–8</sup> Landmark demonstrations such as Ada, which is a self-driving laboratory for thin-film materials optimization, and the mobile robotic chemist for autonomous photocatalyst discovery illustrate both the promise of this paradigm and the dependence of its success on software that couples high-level experimental intent to low-level hardware execution.<sup>9,10</sup>

Within this broader movement, the Aspuru-Guzik group has played a central conceptual role. ChemOS provided an orchestration layer for autonomous experimentation and clarified how planning, execution, and analysis modules can be coupled within a unified software stack.<sup>11</sup> Subsequent work from the same community and related collaborators further defined the scope, opportunities, and practical challenges of self-driving laboratories, including questions of interoperability, autonomy level, cost, and laboratory accessibility.<sup>1,3,5,7</sup> These studies established a systems-level perspective, but they also make clear that overall autonomy depends critically on the reliability of the procedural representation that connects human-readable chemistry with machine execution.<sup>3,5</sup>

<sup>a</sup>Autonomous Polymer Design and Discovery Group, National Institute for Materials Science, 1-2-1 Sengen, Tsukuba, Ibaraki 305-0047, Japan. E-mail: NAGATA.Yuuya@nims.go.jp

<sup>b</sup>Graduate School of Information Science and Technology, Hokkaido University, Kita 14 Nishi 9, Kita-ku, Sapporo, Hokkaido 060-0814, Japan

<sup>c</sup>Advanced General Intelligence for Science Program (AGIS), TRIP Headquarters, RIKEN, Wako, Saitama, Japan

<sup>d</sup>Institute for Chemical Reaction Design and Discovery (WPI-ICReDD), Hokkaido University, Kita 21 Nishi 10, Kita-ku, Sapporo, Hokkaido 001-0021, Japan

<sup>e</sup>Faculty of Information Science and Technology, Hokkaido University, Kita 14 Nishi 9, Kita-ku, Sapporo, Hokkaido 060-0814, Japan

In parallel with these developments, the Cronin group introduced a complementary line of research based on a formal chemical programming language and modular hardware. The Chemputer work showed that synthetic procedures can be decomposed into abstract operations and executed reproducibly in a modular robotic environment.<sup>12</sup> This abstraction was subsequently extended into a universal framework for digitizing and automatically executing the chemical synthesis literature, in which  $\chi$ DL was proposed as a machine-readable and hardware-agnostic representation of synthetic protocols.<sup>13</sup> More recent studies demonstrated cross-platform repeatability and expanded programmability through reusable higher-level constructs such as reaction blueprints and logical control flow.<sup>14,15</sup> Collectively, these studies indicate that  $\chi$ DL can serve as a portable intermediate language between textual chemistry knowledge and robotic execution.

The conversion of textual procedures into structured procedure languages has also advanced substantially in recent years. Vaucher and coworkers at IBM Research trained a transformer-based sequence-to-sequence model that converts English experimental procedures into structured action sequences and deployed it on the cloud-based RXN for Chemistry platform.<sup>16</sup> We previously developed OSPAR, which is a corpus for extraction of organic synthesis procedures with argument roles, to represent actions together with semantically related entities and parameters in an interpretable annotation framework.<sup>17</sup> We then proposed a framework for reviewing the results of automated conversion of structured organic synthesis procedures from the literature, in which annotated text and multiple candidate structured outputs support human correction and validation.<sup>18</sup> These studies addressed an upstream problem, namely how natural-language procedures can be mapped to structured procedural descriptions while preserving semantic transparency for expert review. This upstream perspective has recently been strengthened by work from the Aspuru-Guzik group and collaborators on large language models for chemistry robotics. In particular, CLAIRify showed that natural-language instructions can be translated into robot-executable plans with verifier-assisted generation, using  $\chi$ DL as a domain-specific language in a broader task-and-motion-planning framework.<sup>19</sup>

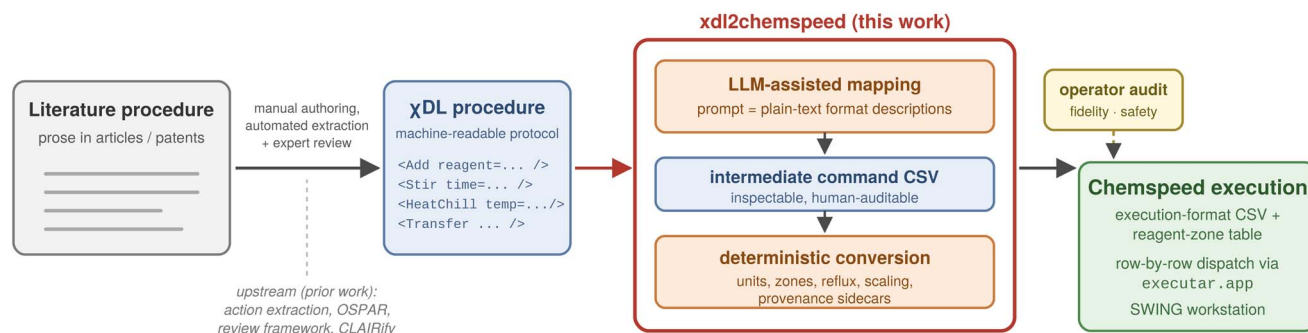
Nevertheless, an intermediate language is only useful in practice when robust translation layers are available for specific experimental infrastructures. Many materials laboratories operate commercial automation systems rather than Chemputer architectures or fully bespoke robotic platforms. In such settings, the practical bottleneck often lies not in the conceptual validity of  $\chi$ DL but in the absence of reliable software that converts  $\chi$ DL procedures into the command structures, reagent tables, zone assignments, and hardware conventions required by a given platform.<sup>5,6,20</sup> This issue is particularly important for Chemspeed systems, which are among the most widely deployed commercial platforms for parallel organic synthesis, formulation, and materials screening. The mismatch in abstraction level between  $\chi$ DL, which is a chemically motivated action language, and the Chemspeed program format, which is

a zone-oriented command table, is the gap that the present work addresses.

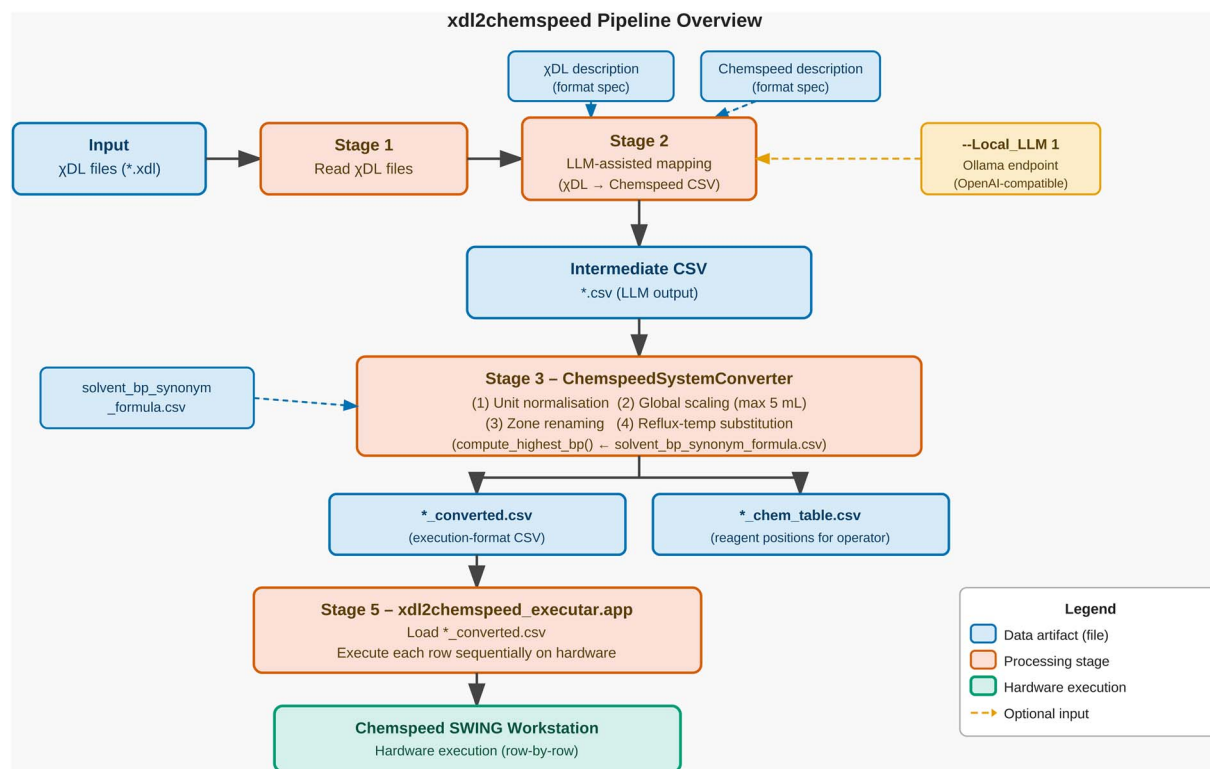
The present study focuses on the deployment layer between structured procedure representation and commercial hardware execution. Rather than generating  $\chi$ DL from text, we assume that an  $\chi$ DL procedure is already available, either by manual authoring or by automated extraction and review.<sup>16–18</sup> We then convert that  $\chi$ DL procedure into a Chemspeed-oriented program with platform-specific normalization and parser-robust formatting. The contribution of this paper therefore lies between literature digitization and robotic execution on a commercial materials-synthesis platform. Fig. 1 places this contribution in the overall workflow from literature prose to hardware execution; the corresponding software architecture is detailed later in Fig. 2.

We deliberately adopt an LLM-assisted mapping stage for this deployment layer, rather than a purely rule-based XML-to-CSV translator. This choice is motivated by three considerations. First, the broader pipeline that this work is designed to serve begins from natural-language experimental procedures. CLAIRify demonstrated that natural-language instructions can be translated into  $\chi$ DL through verifier-assisted LLM generation,<sup>19</sup> and our own review-oriented framework for automated conversion of organic synthesis procedures likewise relies on LLM-based structured-output generation as its core component.<sup>18</sup> Because the intended end-to-end workflow is natural language  $\rightarrow$   $\chi$ DL  $\rightarrow$  Chemspeed, it is architecturally consistent to retain an LLM-based interface at the downstream  $\chi$ DL  $\rightarrow$  Chemspeed step as well, so that the whole chain can share prompt-engineering infrastructure, model endpoints, and operational monitoring. Second,  $\chi$ DL itself is a living specification whose action vocabulary is expected to grow as additional hardware capabilities are incorporated. A prompt-driven mapping stage that consumes a textual format description can accommodate new or rarely encountered  $\chi$ DL elements without re-implementing a deterministic parser, whereas a hand-coded rule table must be extended for every new element. Third, LLMs naturally absorb benign surface variation in the  $\chi$ DL input, such as synonymous parameter names, minor whitespace or casing differences, and non-canonical reagent labels, which would otherwise require extensive defensive coding. We combine this flexibility with deterministic rule-based post-processing (Stage 3), which yielded byte-for-byte identical final machine-readable protocols in the ten-run benchmark described in Section 3.3. We emphasise a practical consequence of this design for prospective users: the mapping stage requires no model fine-tuning and no curated few-shot examples, because its entire behaviour is specified through plain-text format descriptions. Deploying the framework on new or proprietary  $\chi$ DL procedures therefore reduces to editing a text file rather than to training or maintaining a machine-learning model, which substantially lowers the barrier to entry for experimental laboratories that do not employ dedicated machine-learning engineers.

To test whether the  $\chi$ DL  $\rightarrow$  intermediate CSV step in isolation really requires an LLM, we additionally implemented a purely deterministic rule-based converter,



**Fig. 1** Conceptual roadmap of the overall workflow addressed by this study. Published synthesis procedures written as prose are first converted into the machine-readable  $\chi$ DL representation, either by manual authoring or by automated extraction with expert review (prior work: transformer-based action extraction,<sup>16</sup> OSPAR,<sup>17</sup> the review-oriented conversion framework,<sup>18</sup> and CLAIRify<sup>19</sup>). The software reported here (**xdl2chemspeed**, red frame) translates the  $\chi$ DL procedure into an execution-format program: an LLM-assisted mapping stage produces an inspectable intermediate command CSV, and a deterministic converter produces the final workstation-oriented CSV and reagent-zone table together with row-level provenance sidecars. After an operator audit of the generated file, the program is dispatched row by row to the Chemspeed SWING workstation.



**Fig. 2** Software architecture of the **xdl2chemspeed** workflow (see Fig. 1 for the corresponding high-level conceptual roadmap). Blue boxes denote data artifacts, orange boxes denote processing stages, green boxes denote hardware execution, and dashed arrows denote optional inputs. The figure depicts the user-facing conversion and hardware-dispatch pathway from a  $\chi$ DL input file through LLM-assisted mapping, deterministic **ChemspeedSystemConverter** processing, generation of the converted execution-format CSV and chemical-zone table, and row-by-row dispatch to the Chemspeed SWING workstation through **xdl2chemspeed\_executar.app**. The LLM-assisted mapping stage consumes the  $\chi$ DL procedure together with static format descriptions, and either the OpenAI API or a local Ollama endpoint can be selected at the command line. The deterministic converter performs heuristic global scaling against a 5.0 mL-equivalent maximum reactor load, sequential zone renaming, reflux-temperature substitution against a solvent boiling-point table, unit normalisation of mass, volume, and time fields, and final-output provenance propagation. Row-level provenance sidecars are generated as companion audit artifacts during the deterministic reference and downstream-conversion steps. They are intentionally not loaded by the Chemspeed application and are therefore described separately in Table 3 and in the SI.

`xd12chemspeed_rulebased.py`, that parses the  $\chi$ DL input with a standard XML parser and applies the same format-description rules used in the LLM prompt in a dictionary-based dispatcher. The rule-based converter reproduces the same intermediate CSV schema: one-to-one tag mapping of Add, StartStir, Stir, StopStir, HeatChill, HeatChillToTemp, StartHeatChill, StopHeatChill, Wait, Purge, and EvacuateAndRefill; canonical Stir/Wait/StirO triplets for timed Stir steps; normalization of the  $\chi$ DL Hardware element with `id="reactor"` to `reactor1`; the full-transfer total-mixture recovery rule for quantity-omitted Transfer from an unsplit mixture vessel; and the fallback component-level recovery rule when the total-mixture rule is not justified. On the four-case benchmark of Section 5, the intermediate CSVs produced by the rule-based converter are byte-for-byte identical to those produced by the LLM pipeline, and consequently the final `*_converted.csv` files obtained by feeding either intermediate through the same `ChemspeedSystemConverter.py` post-processor are also byte-for-byte identical for all four cases. This cross-implementation equivalence indicates that the Chemspeed format description as currently specified is semantically complete for these procedures and that the LLM stage does not introduce unrecoverable ambiguity relative to a deterministic parser. The rule-based converter is therefore not proposed as a replacement for the LLM stage; rather, it serves as a deterministic reference implementation that cross-validates the LLM output on the benchmark cases and clarifies the role of the LLM in the broader pipeline. The rule-based route also writes a companion provenance file, `<stem>_provenance.csv`, in which each intermediate row is linked to the source  $\chi$ DL file, source-step index, source-step identifier, source-step type, source parameter, original value, normalized value, deterministic conversion rule, software component, and SHA-256 file hashes. When the upstream step is already natural-language  $\rightarrow \chi$ DL by an LLM,<sup>18,19</sup> retaining an LLM at the  $\chi$ DL  $\rightarrow$  Chemspeed step keeps the whole pipeline homogeneous in its prompt-engineering infrastructure, model endpoints, and operational monitoring, and allows the mapping to adapt to future extensions of the  $\chi$ DL action vocabulary through prompt-level updates alone; the rule-based converter would instead require a source-code change for every new tag. For deployments where only the  $\chi$ DL  $\rightarrow$  Chemspeed step is exercised, for example when  $\chi$ DL files are authored or reviewed manually, the rule-based converter offers a deterministic, dependency-free alternative with the same final output and with deterministic source-step provenance. We acknowledge that the LLM route does not remove the maintenance burden entirely: any update of the  $\chi$ DL specification requires re-validation of the mapping layer regardless of whether that layer is implemented as a prompt or as code, and genuinely new command types additionally require extending the downstream execution interpreter. The practical distinction lies in where routine extensions land. Many  $\chi$ DL extensions decompose into the existing Chemspeed command vocabulary (for example, a new action tag whose semantics expand into existing TrsV, Wait, and HeatCool rows), and for these the prompt-based route requires only a format-description edit,

whereas even a well-designed extensible deterministic dispatcher still requires source-code changes, review, and re-release. In addition, whether the LLM route tolerates surface variation of the input that a strict parser rejects is an empirical question, which we examine directly in Section 3.4; the benchmark reported there shows complementary failure surfaces for the two routes under surface perturbation and identifies prompt-level updates that extend the tolerance of the LLM route further. Because the deterministic converter is distributed alongside the LLM route, deployments that prefer a fully deterministic translation path can adopt it directly; the two routes are complementary rather than competing.

## 2 Concept and scope of the software

Because the concrete implementation targets a Chemspeed SWING workstation, it is useful to state at the outset which parts of the framework are generalizable and which are platform-specific. Generalizable components include the separation of the workflow into an inspectable intermediate representation and a final workstation-oriented representation; the prompt-driven mapping stage, whose behaviour is configured entirely by a plain-text format description; the deterministic post-processing stage that absorbs the residual stochasticity of the LLM output; the row-level provenance sidecar schema; and the methodology of cross-validating the LLM route against a deterministic reference converter. Platform-specific components include the concrete command vocabulary and CSV schema of Table 1, the sequential zone-naming convention, the loading-scale heuristic and its parameters, the reflux-substitution table, and the `xd12chemspeed_executar.app` dispatch layer. Porting the framework to a different automation platform therefore amounts to replacing the format description and the emission rules of the downstream converter, while the architecture, prompting strategy, provenance schema, and validation methodology carry over unchanged. Where helpful, the stage descriptions below open with an indication of which side of this division they belong to.

### 2.1 Design principles

The software, hereafter referred to as `xd12chemspeed`, is designed to accept  $\chi$ DL files that represent organic synthesis procedures and to transform them into Chemspeed-oriented command files. The present implementation focuses on the execution layer rather than on the extraction of procedures from raw literature text. The software therefore builds upon the structured representations established in our earlier work<sup>17,18</sup> and on existing upstream extraction pipelines.<sup>16,19</sup>

Five design principles guide the current architecture. First, the workflow is separated into an intermediate representation and a final workstation-oriented representation. This separation improves maintainability and makes the mapping from  $\chi$ DL actions to Chemspeed operations inspectable. Second, downstream formatting is isolated in a dedicated converter script. This design permits local changes in naming, scaling, and reflux handling without modification of the upstream

**Table 1** Chemspeed control-command set used by the present pipeline. Each row of the Chemspeed execution-format CSV is one of these commands. Items in square brackets are optional fields permitted by the format description; the exact CSV-arity and trailing-comma conventions supplied to the LLM are documented in the SI. The VacRefill entry is included for completeness because it can appear in the intermediate CSV as the translation target of  $\chi$ DL EvacuateAndRefill and Purge actions, but in the present deployment it is intentionally omitted by the downstream converter for the deck-level inertisation reason discussed in Sections 2.1 and 4.5

| Command   | Schema (CSV fields)                              | Role on the Chemspeed workstation                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-----------|--------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TrsV      | TrsV, SourceZone, DestinZone, Volume, [Flowrate] | Volumetric liquid transfer from SourceZone to DestinZone. Volume carries an explicit unit (mL or $\mu$ L) in the intermediate CSV and a numerical mL-equivalent value in the converted CSV after global loading scaling                                                                                                                                                                                                                                                          |
| TrsG      | TrsG, SourceZone, DestinZone, Weight             | Gravimetric solid transfer. Weight carries an explicit unit (g or mg) in the intermediate CSV and a numerical g-equivalent value in the converted CSV after global loading scaling                                                                                                                                                                                                                                                                                               |
| Stir      | Stir, Zone, [AngularSpeed]                       | Start stirring in the specified zone. The optional AngularSpeed field carries an explicit rate (e.g. 200 rpm) when provided by the source $\chi$ DL; otherwise it is left blank and the workstation applies its default rate. A timed $\chi$ DL Stir step is expanded into the canonical triplet Stir, Zone/Wait, T/StirOff, Zone                                                                                                                                                |
| StirOff   | StirOff, Zone                                    | Stop stirring in the specified zone. Emitted both as the closing line of the canonical Stir/Wait/StirOff triplet for a timed $\chi$ DL Stir step and as the translation of an explicit $\chi$ DL StopStir action                                                                                                                                                                                                                                                                 |
| HeatCool  | HeatCool, Zone, Temperature                      | Set the temperature setpoint of the specified zone (used for both heating and cooling). Temperature carries an explicit unit ( $^{\circ}$ C, K, or the special token reflux) in the intermediate CSV and a numerical $^{\circ}$ C value in the converted CSV, with reflux resolved against the solvent boiling-point table during Stage 3 conversion. A terminal $\chi$ DL StopHeatChill action is rendered as HeatCool, zone, 25.0000 to return the zone to ambient temperature |
| Wait      | Wait, Time                                       | Hold the current state of the workstation for the specified duration. Time carries an explicit unit (s, min, or h) in the intermediate CSV and is normalised to minutes in the converted CSV                                                                                                                                                                                                                                                                                     |
| VacRefill | VacRefill, Gas, [Iteration]                      | Replacement (evacuate/refill) cycle for a zone, with the specified inert gas (one of Ar, N <sub>2</sub> , CO <sub>2</sub> ). Used as the translation target of $\chi$ DL EvacuateAndRefill and purge actions in the intermediate CSV. Intentionally omitted (schema dropout, Section 5) by the downstream converter in the present deployment (see Sections 2.1 and 4.5)                                                                                                         |
| END       | END                                              | Terminate the program. Each Chemspeed execution-format CSV ends with a single END row                                                                                                                                                                                                                                                                                                                                                                                            |

mapping prompt. Third, the pipeline includes an explicit protection step for comma-containing reagent names. In the current implementation this protection is achieved by selective quotation of only the second CSV field in TrsG (solid transfer) and TrsV (liquid transfer) rows; Table 1 summarises the full Chemspeed control-command set used by the pipeline. This design preserves reagent labels such as 3,4-dimethoxyphenol without breaking CSV column structure. Fourth, row-level provenance is written as sidecar CSV files rather than as additional columns in the execution CSV. This design preserves compatibility with the Chemspeed application while retaining an auditable transformation trace for each generated row. For a chemistry reader, these sidecar files play the same role as a laboratory-notebook cross-reference: for every row that the robot will execute, they answer the questions of which literature-derived step produced it and which rule produced its numerical value. The SHA-256 file hashes recorded in the sidecars in turn guarantee that the file being audited is exactly the file that was, or will be, executed, so that a provenance record cannot silently refer to an outdated or edited copy of the program. Fifth, the  $\chi$ DL  $\rightarrow$  intermediate CSV mapping is delegated to an LLM rather than to a hand-coded deterministic translator in the primary route. This decision keeps the mapping layer homogeneous with the upstream natural-

language  $\rightarrow$   $\chi$ DL stage used in CLAIRify<sup>19</sup> and in our own review-oriented framework,<sup>18</sup> and it allows the mapping to absorb extensions of the  $\chi$ DL vocabulary or unfamiliar action elements simply by updating the accompanying format description, without re-implementing a parser and without any model fine-tuning or curated few-shot examples. The residual formatting variability of the LLM stage is then removed for the benchmark cases by deterministic Stage 3 post-processing, as quantified in Section 3.3.

The present implementation also has explicit limits that are important for interpretation. The deterministic rule-based route provides source-step provenance for the benchmark cases, and the downstream converter propagates this provenance into final-output audit files. The LLM-assisted route writes generated-row metadata but does not claim deterministic source-step attribution, because the prompt-based output is not yet coupled to a verifier that enforces a one-to-one relation between input actions and output rows. For this reason, LLM-derived intermediate files should be treated as reviewable artifacts before execution, even though the rule-based cross-check provides deterministic provenance for the same benchmark outputs. In addition, the converter intentionally maps inert-gas exchange commands such as EvacuateAndRe $\emptyset$ ll to no-op rows because the target Chemspeed SWING workstation in

this study performs an instrument-level inert-gas purge of the entire deck. Under the standard operating conditions used in our laboratory, nitrogen purging of the workstation enclosure reduces the ambient oxygen concentration inside the deck to below 100 ppm as measured by an in-line oxygen sensor, and we have confirmed empirically that a range of oxygen-sensitive organometallic reactions, including palladium- and copper-catalysed cross-couplings, proceed without complication in this atmosphere. Per-vial gas exchange is therefore redundant for the enclosure used here. We emphasise, however, that this is a deployment-dependent design choice: operators using a workstation without enclosure-level inertisation will need to provide an alternative mapping rule that emits explicit per-vial gas-exchange rows into the converted CSV. Rows that are intentionally dropped in the current deployment are retained in the final provenance sidecar with `status=dropped_by_downstream_converter`, so that schema-level omissions remain visible during audit.

## 2.2 Chemspeed program format and the translation gap

A Chemspeed program is a tabular command file in which each row represents a hardware-level operation such as liquid transfer, solid transfer, stirring, heating, waiting, or program termination. Rows reference zones, reagents, and numerical parameters with units that are expected by the downstream workstation environment.  $\chi$ DL, in contrast, is an XML-based action language whose elements, such as `Add`, `HeatChill`, and `Stir`, refer to chemical actions rather than hardware addresses and whose parameters may use different textual conventions. Direct one-to-one mapping is therefore not guaranteed in general. A single  $\chi$ DL `HeatChill` block may correspond to several Chemspeed rows or to a row plus accompanying waiting logic, depending on the target format. The gap between chemically meaningful actions and hardware-oriented command rows motivates the use of an intermediate representation that can be inspected before final conversion. The complete set of Chemspeed control commands used by the pipeline, together with their CSV schemas and roles on the workstation, is given in Table 1; subsequent sections refer to these commands without re-introducing them.

# 3 Software workflow

## 3.1 Implementation details

This subsection describes the concrete realisation of the pipeline: the two-stage architecture itself is platform-agnostic, whereas the file formats and conversion rules described here are specific to the Chemspeed deployment. The developed pipeline consists of a series of conversion and normalization steps, whose software architecture is summarized in Fig. 2 (Fig. 1 gives the corresponding high-level conceptual roadmap). The main script, `xdl2chemspeed.py`, reads `.xdl` files from an input directory and writes generated artifacts to an output directory. The script can connect either to the OpenAI API or to a local Ollama OpenAI-compatible endpoint. In the implementation used for the reported experiments, the default

remote model is `gpt-4o-2024-05-13`, and the default local model is `gpt-oss-120b`; the function-level structure of the client construction, the endpoint selection, and the command-line interface are documented in the SI.

The LLM prompt is assembled entirely as a user message. The prompt concatenates an  $\chi$ DL description file, a Chemspeed format description file, the raw  $\chi$ DL procedure, and an instruction that requires the model to write the generated program between `<OUTPUT>` and `</OUTPUT>` tags. The current script does not introduce a separate system prompt, few-shot examples, automated retries, or a verifier model. The sampling configuration is chosen so that the randomness of the LLM output is controlled solely through the `temperature` parameter and so that the legitimate row-level token repetitions of the target CSV schema are not penalised; the exact parameter values, the rationale for each choice, and the model-dependent token-limit handling are documented in the SI. After inference, the script extracts the text between the output tags and removes blank lines.

A dedicated post-processing step then reconstructs only the second CSV field of `TrsG` and `TrsV` lines and encloses that field in quotation marks (the corresponding function-level details are given in the SI). This step is central to the present study because naive comma splitting would otherwise corrupt reagent labels that contain commas.

The downstream converter, `Chem-speedSystemConverter.py`, reads the intermediate CSV with a standard CSV parser (parser settings in the SI) and applies a heuristic loading model. In the reviewed version supplied with this manuscript, liquid inputs are weighted by a fixed density proxy of 0.80, solid inputs are weighted by a fixed density proxy of 1.00, and the default target maximum zone load proxy is 5.0 mL-equivalent. The resulting global scaling factor is calculated from the largest aggregated zone-load proxy according to eqn (1). This calculation is a practical scheduling heuristic rather than a physical occupancy calculation. It is intended to generate a conservative first-pass Chemspeed program for operator review, not to certify vial fill levels without human inspection. For this reason, the execution-format CSV should be reviewed by the operator before hardware deployment, with particular attention to reaction concentration, mixing, headspace, solid swelling, and installation-specific dispensing limits. The converter generates sequential identifiers such as `zone-1` and `solid-1`. It also resolves the special token `reflux` by searching `solvent_bp_synonym_formula.csv` after removal of the reagent prefix before the first underscore. The synonym-expanded boiling-point table was constructed by taking a base list of common organic solvents and their boiling points (`solvent_bp.csv`, 173 entries) and expanding each entry with synonyms, CAS registry numbers, and common trade or chemical-supplier names retrieved from PubChem;<sup>21</sup> the resulting lookup table contains approximately 16 000 rows and substantially improves matching robustness when reagent labels in the input  $\chi$ DL use non-canonical names. If no matching solvent is found, the converter returns a default value of 80.0000. The reviewed converter also corrects latent unit-conversion errors for mg, uL,

uL/min, and Hz; these units are not used in the archived benchmark CSV outputs, so the correction does not alter the four converted CSV files reported in the manuscript. When a companion intermediate provenance file is present, the converter writes `<stem>_converted_provenance.csv` and propagates source-step metadata, conversion-rule identifiers, zone assignments, unit transformations, the global scaling factor, the software version, the chemical-zone table filename, and file hashes into the final-output audit trail.

### 3.2 Validation scope

The present revision supports one literature-derived case study with hardware execution and three additional literature-derived procedures for file-generation and fidelity benchmarking. It demonstrates parser-level robustness, transparent inspection of intermediate outputs, output-layer reproducibility over ten independent runs (Section 3.3), deterministic row-level provenance for the rule-based reference route, and end-to-end hardware execution on a Chemspeed SWING workstation for the benzylation case. It does not yet provide hardware validation for all benchmark procedures, deterministic source-step attribution for the LLM-assisted route, or blind multi-annotator step-fidelity scoring. Table 2 summarizes the current scope.

The hardware execution result establishes that the automatically generated command file is operationally valid for the present case study. Translation behaviour across a small benchmark of additional literature-derived  $\chi$ DL procedures is

reported in Section 5. Hardware execution of those additional cases, verifier-coupled source-step attribution for the LLM-assisted route, and blind multi-annotator scoring remain future work.

### 3.3 Reproducibility of the LLM-assisted mapping stage

To evaluate the reproducibility of the LLM-based conversion pipeline, ten independent runs were performed using a fixed sampling temperature of 0.2. The raw text returned by the LLM exhibited minor run-to-run variation, restricted to whitespace handling and numerical formatting (for example, “8.32 g” versus “8.32 g”, or the presence of trailing commas) rather than to the structural content, such as row count, command order, or reagent assignment. The deterministic second-field quoting step executed immediately after the LLM call, and the subsequent Stage 3 processing in `Chem-speedSystemConverter.py` (unit normalisation, global scaling, zone renaming, and reflux substitution, with rounding to four decimal places) absorbed these formatting differences completely. As a result, all ten final `*_converted.csv` files produced at the end of Stage 3 were byte-for-byte identical. This combination of low-temperature sampling and rule-based post-processing gave byte-for-byte identical final machine-readable protocols in the archived ten-run benchmark, even though the intermediate CSV remained slightly variable in formatting. The same ten-run reproducibility test was repeated using the local `gpt-oss-120b` model served through the Ollama OpenAI-

Table 2 Validation scope in the present manuscript

| Item                            | Current status                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Source procedures               | One hardware-executed literature-derived $\chi$ DL case study based on Step A of the <i>Organic Syntheses</i> article by Okaya <i>et al.</i> , <sup>22</sup> plus three additional literature-derived $\chi$ DL benchmark procedures drawn from <i>Organic Syntheses</i>                                                                                                                                                                                                                                                                                                                                                              |
| Intermediate generation         | Successful creation of an intermediate CSV file from the $\chi$ DL input using the uploaded prompt-based workflow                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Comma-containing label handling | Successful preservation of reagent labels such as 3,4-dimethoxyphenol through selective quotation of the second field in TrsG and TrsV rows                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Converter behaviour             | Successful creation of <code>v93p0063_converted.csv</code> and <code>v93p0063_chem_table.csv</code> from the intermediate CSV in the supplied case study                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| LLM output reproducibility      | Confirmed over ten independent runs at sampling temperature 0.2. Stage 2 intermediate outputs showed minor run-to-run variation; all ten final CSVs produced by the deterministic Stage 3 post-processor were byte-for-byte identical                                                                                                                                                                                                                                                                                                                                                                                                 |
| Chemspeed hardware execution    | Validated. The generated <code>v93p0063_converted.csv</code> was executed without manual modification on a Chemspeed SWING workstation equipped with a temperature-controlled 8 mL vial rack, a liquid dispensing unit, and a solid dispensing unit. The reaction was performed at 2.65 mmol scale, corresponding to the quantities in the converted CSV after global scaling. After purification by silica gel column chromatography (15% EtOAc/85% <i>n</i> -hexane), the target product was isolated in 84.4% yield (546.12 mg) with <sup>1</sup> H NMR data consistent with the literature (see SI)                               |
| Multi-case fidelity benchmark   | Performed on three additional literature-derived $\chi$ DL procedures (Okamoto Step A, <sup>23</sup> Di Maso Michael-addition step, <sup>24</sup> and Benkovics Step A <sup>25</sup> ) drawn from <i>Organic Syntheses</i> . A retrospective expert-curated step-level fidelity analysis across the case study and these three additional cases (47 $\chi$ DL actions in total) is reported in Table 5 and classifies 44 actions as correctly expanded and 3 as schema dropouts (intentionally omitted because the current Chemspeed schema has no corresponding field), with no action-level mistranslations observed; see Section 5 |
| Row-level provenance tracing    | Implemented for the deterministic XML-parsing reference route and the downstream converter. Intermediate and final sidecars map generated rows to source steps, source parameters, conversion rules, unit normalization, zone assignments, scaling factors, file hashes, and dropped-row status. The LLM-assisted route records generated-row metadata only                                                                                                                                                                                                                                                                           |

compatible endpoint. The local runs were performed on a Windows 11 Pro (64-bit) workstation equipped with an AMD Ryzen Threadripper 3990X processor, 128 GB of DDR4-3200 memory, and an NVIDIA RTX PRO 6000 Blackwell Max-Q Workstation Edition GPU with 96 GB of VRAM, on which the `gpt-oss-120b` model was served through Ollama. The qualitative behaviour was identical to that of `gpt-4o-2024-05-13`: the intermediate CSV exhibited the same class of minor white-space and numerical-formatting variations across runs, while all ten final `*_converted.csv` files produced after the deterministic Stage 3 post-processing were byte-for-byte identical. These observations lead us to believe that output-layer determinism for the benchmark cases is not specific to the remote commercial model and that an on-premises open-weight model can be substituted without loss of determinism in the final converted CSV files, although we emphasise that this behaviour has been verified only for the four benchmark procedures and the two models examined here. This is a practically important observation for laboratories that cannot transmit synthetic procedure data to an external API.

### 3.4 Robustness to input surface variation

One motivation for the LLM-assisted mapping stage stated in the Introduction is tolerance of benign surface variation in the  $\chi$ DL input. To test this property directly rather than assert it, we constructed a perturbation benchmark from the four benchmark  $\chi$ DL files. Each file was subjected to three classes of deterministic, seed-controlled perturbations that preserve the chemical meaning of the procedure: (i) formatting perturbations, in which the XML is re-indented or collapsed onto fewer lines and the attribute order of the action elements is permuted; (ii) lexical perturbations, in which unit spellings and value spacing are varied (for example, `min`  $\rightarrow$  `minutes`, `h`  $\rightarrow$  `hours`, `mL`  $\rightarrow$  `mL`, and removal of the space between a number and its unit); and (iii) typographical perturbations, in which realistic character-level typos are introduced into reagent labels that do not participate in reflux-temperature resolution (for example, `potassium carbonate`  $\rightarrow$  `potassium carbonate`). All three classes leave the correct final CSV unchanged: for classes (i) and (ii) the canonical output is independent of the serialization details and of the surface spelling of the units (restoring the canonical unit spellings is the task of the mapping stage, since the Stage 3 converter accepts only the canonical forms), and for class (iii) only the reagent labels in the human-readable chemical-zone table legitimately differ while the execution CSV rows are unchanged. Two variants per class were generated for each of the four cases (24 perturbed inputs in total). Each perturbed file was passed through the complete LLM-assisted pipeline (`gpt-4o-2024-05-13`, temperature 0.2), and the resulting final `*_converted.csv` was compared with the corresponding unperturbed reference output both byte-for-byte and at the level of parsed command sequences and numerical fields. The perturbation generator, the evaluation script, and the perturbed inputs are included in the software archive (`robustness_eval/`), and the protocol is described in the SI.

Across the 24 perturbed variants, 15/24 final `*_converted.csv` files were byte-for-byte identical to the unperturbed reference outputs (typographical class: 8/8; formatting class: 6/8; lexical class: 1/8), and the same 15 were identical at the parsed command-sequence and numerical-field level. The nine deviations fall into two well-defined groups, neither of which silently altered the chemistry of an executable file. Seven of the eight lexical variants were not normalised by the model: the perturbed unit surface forms (for example 16 hours and 8.32 g) were echoed verbatim into the intermediate CSV, and the deterministic Stage 3 converter then rejected each of these files with an explicit unit-parsing error, so that these variants fail loudly at conversion instead of producing an executable program; unit normalisation was not consistent across cases, since the remaining lexical variant (the spelled-out units of `v97p0054`) was normalised correctly and passed byte-for-byte. The other two deviations are the formatting variants of `v97p0054`, in which the dual-quantity Add of diphenylphosphinic chloride (the source element carries both `mass="50.0 g"` and `volume="40.4 mL"`; Section 5) is emitted through the gravimetric `TrsG` form instead of the volumetric `TrsV` form of the reference run, the two encodings being equivalent in reagent amount and both admitted by the format description. Row-by-row inspection of the archived variant outputs shows that the deviation is confined to the encoding of this single addition and to the consequent renumbering of the side-flask zone label in the rows that reference it (the reagent is registered as `solid-2` rather than as a liquid reservoir zone, which shifts the sequential zone numbering by one): every dispensed quantity, hold time, and temperature setpoint, the recovered full-transfer volume of the side-flask contents, and the global scaling factor are identical to the reference. Because the affected reagent is a liquid, the gravimetric encoding is operationally dispreferred, and this visible command-level change is exactly the class of deviation that the operator audit of Section 3.5 is designed to catch. The benchmark complements the ten-run reproducibility analysis above: reproducibility fixes the input and varies the sampling, whereas the present benchmark fixes the sampling regime and varies the input surface.

As a deterministic point of comparison, feeding the same 24 perturbed inputs through the rule-based reference converter (`xdl2chemspeed_rulebased.py`) and the same downstream converter yields 16/24 byte-for-byte identical final CSVs: all formatting and typographical variants are absorbed, because the XML parser is indifferent to serialization details and the zone assignment is independent of the label text, whereas all eight lexical variants fail during downstream conversion, because the deterministic route copies every unit surface form verbatim. Under the strict byte-for-byte criterion the two routes therefore perform comparably on this benchmark (15/24 *versus* 16/24), with complementary failure surfaces: the deterministic route is immune to formatting perturbation but structurally incapable of absorbing lexical variation, whereas the LLM route absorbed one lexical variant, echoed the remainder, and introduced one admissible encoding flip under formatting perturbation. Both observed LLM-route shortfalls are addressable at

prompt level, which is the extensibility mechanism emphasised throughout this work: an explicit instruction in the Chemspeed format description to normalise unit spellings to the canonical forms, and an explicit precedence rule for dual-quantity Add elements (prefer the volumetric form when both mass and volume are present), are one-line format-description updates that require no code change, and verifying their effect on this benchmark is identified as an immediate next step. Equally important is the fail-safe character of the observed deviations: in no variant did the pipeline silently produce an executable file with altered chemistry—the lexical failures abort at Stage 3 with explicit errors, and the encoding flip is a visible command-type change caught by the audit. The archived evaluation reports for both routes are included in the software archive.

### 3.5 Hardware-configuration compatibility and operator review

Compatibility between a generated workflow and a specific hardware configuration is established at three levels in the present framework. First, the Chemspeed format description that accompanies the prompt enumerates only the command types that the deployed configuration supports (Table 1); the capability set of the installation is thus encoded in the format description itself, and a procedure that requires operations outside this set cannot be expressed in the generated program—the corresponding actions surface as schema dropouts that are retained in the provenance sidecar rather than being emitted as unexecutable rows. Second, the physical addressability of the deck is fixed by a static template, as described in the next paragraph. Third, no automated model of module availability, accessory configuration, or operational capability is currently implemented: the final compatibility judgement is delegated to an operator audit of the generated file.

The logical identifiers assigned by the converter (*zone-1*, *solid-1*, and so on) are bound to physical deck coordinates through a static template defined inside `xd12chemspeed_executar.app`: each logical label corresponds to a fixed rack position of the deployed deck layout (Fig. S2 of the SI). Dynamic physical auto-zoning is not supported in the present implementation. Adapting the pipeline to a different deck layout or rack population is therefore a once-per-deployment edit of this template rather than a per-procedure operation.

Beyond syntactic conformance, physical executability is not certified automatically. Constraints such as viscous-liquid dispensing, foaming, solid flowability in the gravimetric dispensing unit, vial headspace, and dead volumes are outside the scope of the loading heuristic of eqn (1), which is deliberately conservative rather than predictive. The optional flow-rate field of *TrsV* rows provides a manual mechanism for slowing the dispensing of viscous liquids, but its value is currently assigned by the operator rather than inferred from reagent properties; automated property-aware parameterisation of dispensing operations is identified as future work.

We therefore emphasise that the transition from the generated CSV to hardware execution is deliberately not fully automated. An expert operator audits the generated file for chemical

fidelity and physical safety (fill volumes, dead volumes, dispensing limits, and reagent compatibility) before it is loaded into `xd12chemspeed_executar.app`. The statement that the case-study program was executed “without manual modification” (Section 4.5) therefore means that the operator audit did not necessitate any edit of the generated file, not that the audit step was skipped. The pipeline automates the translation of the procedure, not the authorisation of its execution.

## 4 Case study: benzylation of 3,4-dimethoxyphenol

To illustrate the behaviour of `xd12chemspeed` on a realistic literature procedure, we applied the complete pipeline to the synthesis of 4-benzyloxy-1,2-dimethoxybenzene reported as Step A in an *Organic Syntheses* article by Okaya, Okuyama, Okano, and Tokuyama.<sup>22</sup> The article describes charging 3,4-dimethoxyphenol and potassium carbonate, adding MeCN, stirring at ambient temperature, adding benzyl bromide, washing the reflux condenser with MeCN, and then heating the reaction mixture to reflux for 2 h. This fragment is well suited to the present proof-of-concept because it contains comma-bearing reagent names, both solid and liquid additions, a repeated use of the same solvent, and a reflux instruction that must be translated into a workstation-oriented temperature field (Fig. 3).

### 4.1 Stage 1: $\chi$ DL input

The case-study input file `v93p0063.xd1` declares four reagent entries, one reactor, and four source flasks (one per reagent). Each reagent is given an internal identifier of the form *ri\_name* (where *i* is the reagent index in the  $\chi$ DL *Reagents* block); these prefixed identifiers propagate through the intermediate CSV and the chemical-zone mapping table. The *Procedure* block contains eleven actions: two *Add* steps for solids, two *Start-Stir* declarations, two *Stir* steps with explicit times, two liquid *Add* steps for MeCN, one liquid *Add* step for benzyl bromide, one *HeatChill* step, and one *StopStir* step. Two points are important for accurate interpretation. First, the  $\chi$ DL input already contains the 5 mL MeCN rinse as an explicit *Add* action. Second, the *HeatChill* block already uses *temp* = “*reflux*” rather than a numerical temperature. The present case study therefore tests preservation and downstream handling of these instructions rather than LLM recovery of omitted information.

### 4.2 Stage 2: LLM-assisted mapping to intermediate CSV

The mapping strategy of this stage is generalizable; only the target command vocabulary is Chemspeed-specific. The  $\chi$ DL file was passed to the LLM stage together with the static  $\chi$ DL and Chemspeed format descriptions. The resulting intermediate CSV contained seventeen non-empty lines including the terminal *END* row, as reproduced in Listing 1. The output exhibits three properties that are relevant to the present manuscript. First, both 3,4-dimethoxyphenol and potassium carbonate were mapped to the solid-transfer command *TrsG*, whereas MeCN and benzyl bromide were mapped to the

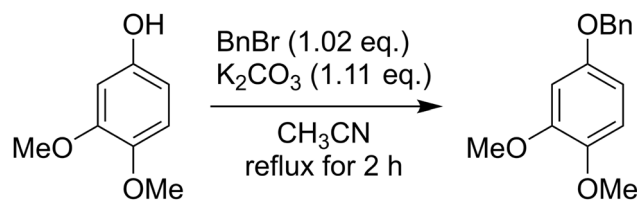


Fig. 3 Reaction scheme for the case study that was adapted from the *Organic Syntheses* source article.

volumetric transfer command `TrsV`. Second, the 5 mL MeCN rinse appears in the intermediate CSV because it is present explicitly in the  $\chi$ DL input. Third, the comma-containing label `r1_3,4-dimethoxyphenol` is retained as a quoted second field after the dedicated post-processing step. The current intermediate format does not preserve an explicit source-action identifier for each generated row, which is an important limitation for future benchmarking. The Chemspeed format description further specifies that a quantity-omitted  $\chi$ DL `Transfer` from an unsplit mixture vessel is recovered as the full-transfer total-mixture volume (the sum of all explicit liquid Add volumes plus solid-contribution volumes at an assumed density of 1 g mL<sup>-1</sup>), and that component-level recovery from a single Add with both volume and mass or amount is used only as a fallback when the total-mixture estimate is not justified. This convention is consistent with the natural chemical reading of a `<Transfer>` without an explicit transferred quantity and is exercised directly in cases v92p0328 and v97p0054 of Section 5.

Listing 1 intermediate Chemspeed CSV after LLM mapping (v93p0063.csv).

```
TrsG,"r1_3,4-dimethoxyphenol", reactor1, 8.32 g
TrsG,"r2_potassium carbonate", reactor1, 8.29 g
Stir, reactor1,
TrsV,"r3_MeCN", reactor1, 95 mL,
Stir, reactor1,
Wait, 10 min
StirOff, reactor1
TrsV,"r4_benzyl bromide", reactor1, 6.54 mL,
TrsV,"r3_MeCN", reactor1, 5 mL,
Stir, reactor1,
Wait, 20 min
StirOff, reactor1
Stir, reactor1,
HeatCool, reactor1, reflux
Wait, 2 h
StirOff, reactor1
END
```

The generated rows do not include optional flow-rate values, and the `Stir` rows leave the optional rotational-speed field blank. These omissions are permitted by the Chemspeed format description that accompanies the prompt. In the present implementation, correctness at this stage therefore means conformance to the expected row schema and faithful retention of the principal additions, waiting periods, and heating instruction from the  $\chi$ DL input.

It is also instructive to compare the eleven  $\chi$ DL actions in the input with the sixteen non-terminal rows of Listing 1. The  $\chi$ DL input contains two explicit `StartStir` declarations and two timed `Stir` steps, and the intermediate CSV therefore contains four distinct `Stir` rows (rows 3, 5, 10, and 13 of Listing 1), in accordance with the format-description rule that each

`StartStir` and each timed `Stir` must emit its own dedicated `Stir` row rather than be absorbed into a neighboring event. Specifically, row 3 maps the first `StartStir`, rows 5–7 form the canonical `Stir/Wait/StirOff` triplet that expands the first timed `Stir` (10 min hold), rows 10–12 form the equivalent triplet for the second timed `Stir` (20 min hold), and row 13 maps the second `StartStir` that precedes the `HeatChill` step. The three terminal `StirOff` rows (rows 7, 12, and 16) correspondingly close the two timed episodes and the final `StopStir` tag. This row-level expansion is consistent with the format-description rule and is captured by the deterministic provenance sidecar of the rule-based reference route. In that sidecar, rows 5–7 of the intermediate CSV are linked to the same source `Stir` step, with distinct conversion-rule identifiers for the start, wait, and terminal `StirOff` rows. The LLM-assisted route still requires operator inspection because it does not yet enforce deterministic source-step attribution without the rule-based cross-check or a verifier-coupled alignment module.

### 4.3 Stage 3: system-specific conversion

Stage 3 is the most platform-specific stage of the pipeline: the parameters below encode properties of the deployed workstation rather than of the  $\chi$ DL representation. The intermediate program was then passed to `ChemSpeedSystemConverter.py`. In the uploaded implementation, the converter uses a maximum reactor load of 5.0 mL-equivalent, an assumed liquid density of  $\rho_L = 0.80$  g mL<sup>-1</sup>, and an assumed solid density of  $\rho_S = 1.00$  g mL<sup>-1</sup>. The converter first aggregates all material directed to each destination zone into a common mass equivalent and then converts that aggregated mass to a volume equivalent through the assumed solid density. For a solid charge of mass  $m_i$  (g), the contribution to the aggregated mass is  $m_i$  itself. For a liquid charge of volume  $V_j$  (mL), the contribution is  $V_j\rho_L$  (g). The total loading for a given destination zone is therefore

$$V_{\text{zone}} = \frac{\sum_{i \in \text{solids}} m_i + \sum_{j \in \text{liquids}} V_j \rho_L}{\rho_S}, \quad (1)$$

so that  $V_{\text{zone}}$  has units of millilitres. The division by  $\rho_S = 1.00$  g mL<sup>-1</sup> is numerically trivial in the present implementation, but it is retained explicitly in eqn (1) to make the dimensional reduction unambiguous. Throughout this paper we report the intermediate sum in the equation above in grams and the final  $V_{\text{zone}}$  in millilitres, which differ only in numerical label for the present choice of  $\rho_S$ . For the reactor in the present case study,

$$\begin{aligned} V_{\text{zone}} &= \frac{8.32 + 8.29 + 95.0 \times 0.80 + 6.54 \times 0.80 + 5.0 \times 0.80 \text{ g}}{1.00 \text{ g mL}^{-1}} \\ &= 101.842 \text{ mL}, \end{aligned} \quad (2)$$

where the first two terms are the solid masses of 3,4-dimethoxyphenol and potassium carbonate, and the remaining three terms are the liquid charges of MeCN (two additions) and benzyl bromide, each converted to a mass equivalent through  $\rho_L$ . The resulting scaling factor is

$$s = \frac{5.0}{101.842} = 0.049096... \approx 0.0491. \quad (3)$$

The converter retains the unrounded value internally; the rounded form 0.0491 is shown here only for readability, and all numerical values reported below are obtained from the full-precision factor. This factor is applied uniformly to the mass and volume fields. In the present code path, the time fields are converted to minutes and are not scaled. Thus, 10 min remains 10.0000, 20 min remains 20.0000, and 2 h becomes 120.0000. The converter assigns sequential identifiers on first use, for example the reactor zone identifier `reactor1` produced by the LLM mapping stage (the normalized form of the  $\chi$ DL Hardware element with `id="reactor"`) becomes `zone-1` in the final CSV, and `r1_3,4-dimethoxyphenol` becomes `solid-1`. Finally, the token `reflux` is replaced by the highest boiling point among matching solvent names found in `solvent_bp_synonym_formula.csv`. In the present case, the matching solvent is MeCN, which gives 81.6 °C.

This stage should be interpreted carefully. The scaling model defined by eqn (1) is a pragmatic loading heuristic with two fixed effective densities and a single maximum-volume threshold. It is not a rigorous model of slurry rheology, dead volume, heat transfer, concentration dependence, or reactor-specific engineering constraints. In particular, the assumed solid density of  $\rho_s = 1.00 \text{ g mL}^{-1}$  is conservative for typical organic solids but underestimates the true density of inorganic bases such as  $\text{K}_2\text{CO}_3$  ( $\sim 2.43 \text{ g mL}^{-1}$ ), so the heuristic intentionally over-allocates effective volume to such reagents in order to stay safely below the vial capacity. The liquid density of  $\rho_L = 0.80 \text{ g mL}^{-1}$  plays the analogous role for commonly used organic solvents. The corresponding execution-format CSV should therefore be regarded as a machine-oriented draft that requires operator review before deployment. Regarding the scale change itself, scale-down from a literature procedure is well established in synthetic chemistry practice: whereas scale-up requires careful attention to mixing efficiency, thermal uniformity, and concentration gradients, scale-down generally improves these parameters and therefore poses fewer risks to reaction fidelity. The experimental outcome described in Section 4.5 supports this reasoning.

#### 4.4 Stage 4: final Chemspeed program and chemical table

The output of the full pipeline consists of the execution files consumed by the Chemspeed workflow and separate audit files. The converted execution-format CSV (`v93p0063_converted.csv`) is shown in Listing 2, and the accompanying chemical table (`v93p0063_chem_table.csv`) is shown in Listing 3. Together, these files provide a workstation-oriented sequence of commands and a human-readable mapping from internal zone identifiers to the original reagent labels. The sidecar files `v93p0063_provenance.csv` and `v93p0063_converted_provenance.csv` provide the row-level audit trail and are not imported into the Chemspeed application.

Listing 2 final system-specific Chemspeed CSV (`v93p0063_converted.csv`).

```
TrsG, solid-1, zone-1, 0.4085
TrsG, solid-2, zone-1, 0.4070
Stir, zone-1,
TrsV, zone-2, zone-1, 4.6641,
Stir, zone-1,
Wait, 10.0000
StirOff, zone-1
TrsV, zone-3, zone-1, 0.3211,
TrsV, zone-2, zone-1, 0.2455,
Stir, zone-1,
Wait, 20.0000
StirOff, zone-1
Stir, zone-1,
HeatCool, zone-1, 81.6000
Wait, 120.0000
StirOff, zone-1
END
```

Listing 3 chemical zone mapping table (`v93p0063_chem_table.csv`).

```
solid-1, "r1_3,4-dimethoxyphenol"
solid-2, "r2_potassium carbonate"
zone-1, "reactor1"
zone-2, "r3_MeCN"
zone-3, "r4_benzyl bromide"
```

A few numerical checks are instructive. The first `TrsG` row corresponds to  $8.32 \times 0.049096 \approx 0.4085 \text{ g}$ . The first `TrsV` row corresponds to  $95 \times 0.049096 \approx 4.6641 \text{ mL}$ . The rinse step corresponds to  $5 \times 0.049096 \approx 0.2455 \text{ mL}$ . The waiting period of 2 h becomes 120 min in the converted file. The chemical table also shows that the original comma-containing reagent label is preserved as quoted text in the human-readable mapping file while the internal identifier used by the converted CSV is `solid-1`. These observations support the claim that the dedicated field-protection step addresses a concrete parsing problem in the selected case study.

The final provenance sidecar provides a row-by-row explanation of the transformation path from source  $\chi$ DL to the device-oriented CSV. Table 3 shows selected fields from the record for the first MeCN transfer in Listing 2. The full sidecar retains additional fields, including the output-field JSON, intermediate-field JSON, source-XDL, intermediate-file, and output-file SHA-256 hashes, the software version, the chemical-zone table filename, and the name of the intermediate provenance file.

The converted execution-format CSV produced at the end of Stage 3 is subsequently loaded into `xd12chemspeed_executor.app`, a companion application implemented in the Chemspeed AutoSuite environment (AutoSuite version 2.29.1.1) that interfaces directly with the Chemspeed hardware. The application reads the CSV file and dispatches each command row to the workstation sequentially in the order in which it appears in the file, translating each row into the corresponding hardware-level instruction (solid transfer, liquid transfer, stirring, heating, waiting, or program termination) *via* the Chemspeed instrument control layer. This row-by-row sequential execution model ensures that the reaction sequence encoded in the generated CSV is faithfully reproduced on the physical workstation without requiring further manual reformatting of the output file. The binding of the logical zone identifiers of the converted CSV to physical deck positions inside this application follows the static template described in Section 3.5.

Table 3 Selected fields from the row-level provenance record for output row 4 of v93p0063\_converted.csv

| Field                           | Recorded value                                                                                                                           |
|---------------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| output_row_id                   | 4                                                                                                                                        |
| output_command                  | TrsV                                                                                                                                     |
| intermediate_row_id             | 4                                                                                                                                        |
| source_step_id                  | v93p0063:step:0004                                                                                                                       |
| source_step_type                | Add                                                                                                                                      |
| source_parameter                | Volume                                                                                                                                   |
| source_value                    | 95 mL                                                                                                                                    |
| intermediate_conversion_rule_id | hdl_add_volume_to_trsv                                                                                                                   |
| conversion_rule_id              | intermediate_trsv_to_chemspeed_trsv; source_zone_assignment; destination_zone_assignment; mL_to_mL_with_global_loading_scale; empty_rate |
| original_value                  | source=r3_MeCN; destination=reactor1; volume=95 mL; rate=                                                                                |
| converted_value                 | source=zone-2; destination=zone-1; volume=4.6641; rate=                                                                                  |
| scaling_factor                  | 0.04909565798                                                                                                                            |
| Status                          | Ok                                                                                                                                       |

#### 4.5 Stage 5: hardware execution

To validate end-to-end operation beyond file generation, we loaded v93p0063\_converted.csv into xdl2chemspeed\_executar.app and executed the program on a Chemspeed SWING workstation equipped with a temperature-controlled 8 mL vial rack, a liquid dispensing unit, and a solid dispensing unit. The application dispatched each CSV row in sequence to the workstation hardware without any further manual modification of the output file; in line with the policy of Section 3.5, the generated file was audited by an operator before execution, and the audit did not necessitate any edit. The reaction was carried out at 2.65 mmol scale with respect to 3,4-dimethoxyphenol, corresponding to the quantities in the converted CSV after global scaling. The Chemspeed enclosure used in this study performs an instrument-level inert-gas purge of the entire deck rather than a per-vial EvacuateAndRefill cycle. Under the standard nitrogen-purge procedure used in our laboratory, the residual oxygen concentration inside the enclosure is reduced to below 100 ppm as measured by an in-line oxygen sensor, and we have separately confirmed that oxygen-sensitive organometallic transformations including palladium- and copper-catalysed cross-couplings proceed cleanly in this atmosphere. The converted CSV therefore does not need to encode per-vial gas-exchange commands for reactions of moderate air sensitivity in this enclosure (see also Section 5).

After the automated run, the reaction mixture was cooled to ambient temperature and filtered through a 2 cm pad of Celite. The workup and purification steps described here were performed off-deck; the status of on-deck filtration and online analysis in the present implementation is discussed in Section 4.6. The transfer was completed by washing the reaction vial with EtOAc (2 × 25 mL). The combined filtrate was concentrated on a rotary evaporator under reduced pressure and dried *in vacuo*, then purified by silica gel column chromatography using 15% ethyl acetate/85% *n*-hexane as the eluent, following the procedure described in the original *Organic Syntheses* article.<sup>22</sup> Thin-layer chromatographic analysis at the end of the reaction confirmed complete consumption of the starting material. The target product, 4-benzyloxy-1,2-

dimethoxybenzene, was isolated in 84.4% yield (546.12 mg from a theoretical 647.4 mg). The <sup>1</sup>H NMR spectrum was consistent with the literature values (see SI for the full spectrum).

The reported yield in the main procedure of the original article is 90% at the 54 mmol scale used in the original *Organic Syntheses* preparation, and Note 7 of the same article additionally reports a half-scale (27 mmol) run that afforded the product in 86% isolated yield (5.70 g).<sup>22</sup> The 84.4% (546.12 mg) isolated yield obtained here at 2.65 mmol scale is therefore comparable to the half-scale literature value documented in Note 7, indicating that the present automated execution reproduces the literature yield within the expected scale-dependent range. TLC confirmed complete consumption of starting material, and the small remaining difference relative to the half-scale literature value is attributable to scale-dependent recovery losses during workup and purification of sub-gram samples, including material retained on the Celite filter pad, residual material in the small reaction vial and on transfer pipettes, and recovery losses during silica gel chromatography. These factors are not consequences of the automated synthesis step itself. These results confirm that the pipeline generates an operationally valid Chemspeed program from a literature χDL file without requiring any manual correction of the output CSV.

#### 4.6 Discussion of the case study

The case study demonstrates end-to-end operation of the present pipeline on a published synthetic procedure, from χDL input through intermediate CSV generation, system-specific conversion, and physical execution on a Chemspeed SWING workstation. The selective quotation step resolves a concrete CSV-formatting problem for reagent labels that contain commas. The hardware execution confirms that the generated program is operationally valid without manual modification.

At the same time, the case study makes the present limitations clear. First, although Section 5 extends file-generation testing to four literature-derived χDL files and reports an expert-curated step-level fidelity analysis in Table 5, only one of these has been validated by physical hardware execution. The

depth/breadth division adopted here, in which one case is executed on hardware and three additional cases are evaluated for translation fidelity against the source  $\chi$ DL, allows the manuscript to demonstrate both operational validity and broader operation-class coverage within a single proof-of-concept. A larger corpus benchmarked on hardware across multiple workstations remains a natural target for future work. Second, the LLM stage is prompt-based and is not coupled to an automated verifier that enforces one-to-one alignment between  $\chi$ DL actions and output rows. The current case did not require recovery of omitted operations because the  $\chi$ DL file already contained the rinse step and the `reflux` token, and the four-case benchmark in Section 5 likewise did not uncover action-level mistranslations; nevertheless, a verifier-coupled LLM variant that flags missing or spurious rows against the  $\chi$ DL would make this guarantee explicit rather than observed, which becomes increasingly important as the benchmark is extended to larger corpora. The deterministic rule-based route now provides row-level provenance for the benchmark outputs, but blind multi-annotator scoring against independently constructed reference CSV files remains necessary before these sidecars can be used as a statistical fidelity benchmark. Third, the isolated yield of 84.4% (546.12 mg) obtained at the present 2.65 mmol scale is comparable to the half-scale (27 mmol) literature yield of 86% (5.70 g) reported in Note 7 of the original *Organic Syntheses* article,<sup>22</sup> while the main-text 54 mmol procedure reports 90%; the present execution therefore reproduces the literature yield within the expected scale-dependent range, and TLC confirmed complete consumption of starting material. For these reasons, the present manuscript should be interpreted as a proof-of-concept with transparent implementation detail, deterministic row-level provenance for the reference conversion route, a quantitative step-level fidelity audit of the translation layer across four literature cases, and experimental confirmation of operational validity for the selected case study.

A related question is how the pipeline behaves when the input procedure is incomplete, ambiguous, or only implicitly specified. The present architecture takes the position that chemical ambiguity should be resolved upstream of the execution layer: the  $\chi$ DL input is assumed to have been authored or reviewed by an expert, for example within our review-oriented conversion framework,<sup>18</sup> so that `xd12chemspeed` is not required to guess missing chemistry. Within this scope, the pipeline handles the following classes of underspecification deterministically. Quantity-omitted `Transfer` actions are recovered by the full-transfer total-mixture rule described in Section 4.2. Optional fields such as stirring rate and flow rate are left blank, so that documented workstation defaults apply. A `reflux` token whose solvent cannot be matched in the boiling-point table falls back to a documented default of 80.0 °C, and this fallback is visible in the provenance record. The known limitations and failure cases of the current architecture are, correspondingly:  $\chi$ DL elements that lie outside the format description are not emitted into the execution CSV (they are retained as schema dropouts in the provenance sidecar); operations that are genuinely absent from the  $\chi$ DL input are not invented by the pipeline, by design; and semantic ambiguity

that survives upstream review propagates into the generated program and must be caught by the operator audit of Section 3.5. The robustness benchmark of Section 3.4 quantifies tolerance to surface-level noise in the input; semantic incompleteness, in contrast, remains out of scope for automated recovery in the present implementation.

The present implementation also deliberately restricts the command vocabulary to synthesis operations. Although the workstation configuration used in this study includes solid-phase-extraction/filtration modules and an online HPLC, we limited the translated operation classes to dispensing, stirring, temperature control, and timed holds in order to keep the translation layer and its validation transparent; the workup and purification of the case study were accordingly performed off-deck (Section 4.5). Extending the Chemspeed format description with filtration, phase-separation, and online-analysis commands, together with the corresponding  $\chi$ DL mappings and hardware validation, is a concrete next step that the two-stage architecture supports without structural change: a new command type requires one new row schema in the format description and one new branch in the executor task tree.

## 5 Multi-case benchmark

Before turning to the individual cases, we clarify the epistemic status of this benchmark. The role of physical hardware execution in the present study is to establish that the automatically generated Chemspeed program is operationally valid on the target workstation. Operational validity means that the pipeline produces a file which, when dispatched row by row by `xd12chemspeed_executar.app`, runs to completion on a real Chemspeed SWING deck and affords the expected product without manual editing. This role is discharged by the benzylation case study of Section 4, in which the execution-format CSV was consumed by the instrument-control layer without any manual editing and produced the target compound with spectroscopic data consistent with the literature. Repeating a full hardware run for every additional  $\chi$ DL input would be informative for inter-batch reproducibility of the workstation hardware itself, but it does not probe a distinct property of the translation pipeline: once the workstation has been shown to accept and execute a pipeline-generated CSV for a non-trivial literature procedure, further validation effort is most informative when it is directed at translation fidelity rather than at re-confirming hardware compliance.

The complementary role of the four-case benchmark is therefore to expose the pipeline to the broader operation-class spectrum that a realistic Chemspeed workflow must support: multi-reagent solid charges, cross-vessel transfers, sub-ambient temperature setpoints, explicit stirring-rate control, and optional  $\chi$ DL annotations whose status in the current Chemspeed schema we can then make transparent. This division of roles between depth (hardware validation on one case) and breadth (file-generation fidelity across four cases) is consistent with standard practice in proof-of-concept translation frameworks for synthesis automation<sup>13,19</sup> and is well matched to the aim of the present manuscript. To the three additional

**Table 4** Multi-case benchmark coverage. A check mark indicates that the operation class is present in the corresponding  $\chi$ DL input and was faithfully reproduced in the generated execution-format CSV. An em dash indicates that the operation class does not occur in that input; it does not indicate a missing or unfaithful reproduction. Every operation class that was present in an input was reproduced in the corresponding output

| Operation class                                   | v93p0063 (ref. 22) | v91p0027 (ref. 23) | v92p0328 (ref. 24) | v97p0054 (ref. 25) |
|---------------------------------------------------|--------------------|--------------------|--------------------|--------------------|
| Solid transfer (TrsG), $\geq 2$ reagents          | ✓                  | ✓                  | ✓                  | —                  |
| Liquid transfer (TrsV)                            | ✓                  | ✓                  | ✓                  | ✓                  |
| Comma-bearing reagent label ( <i>ri</i> _3,4,...) | ✓                  | ✓                  | —                  | —                  |
| Heat above ambient (HeatCool)                     | ✓ (reflux)         | ✓ (60 °C)          | ✓ (110 °C)         | —                  |
| Sub-ambient HeatCool                              | —                  | —                  | —                  | ✓ (−10 °C)         |
| Explicit stir rate (rpm)                          | —                  | —                  | —                  | ✓ (200)            |
| Timed stir → Stir/Wait/StirOff expansion          | ✓                  | ✓                  | ✓                  | ✓                  |
| Cross-vessel transfer (TrsV between zones)        | —                  | —                  | ✓                  | ✓                  |
| Multiple destination zones                        | —                  | —                  | ✓                  | ✓                  |

literature-derived  $\chi$ DL procedures drawn from *Organic Syntheses* we therefore apply the complete `xd12chemspeed` workflow and evaluate the outputs as file-generation and parser-level fidelity checks rather than as independent experimental validations. The four cases collectively cover the operation classes summarised in Table 4.

### 5.1 Step-level fidelity analysis

Table 4 confirms that the pipeline reaches each of the targeted operation classes. To provide a quantitative complement to the deterministic row-level provenance sidecars now generated by the reference conversion route, we additionally performed a retrospective expert-curated step-level fidelity analysis over all four cases. The analysis used only artifacts that the pipeline itself produces (the intermediate CSV, the final `*_converted.csv`, and the chemical-zone mapping table) together with the original  $\chi$ DL input; no additional experimental measurement was required. For each case, the authors manually compared the source  $\chi$ DL procedure with the final execution-format CSV, and classified every  $\chi$ DL action into one of three categories: (i) correctly expanded into the Chemspeed row, triplet, or cross-vessel transfer specified by the format description; (ii) a schema dropout, meaning that the action was translated correctly but is intentionally not emitted into the final execution CSV because the current Chemspeed format description does not expose a field capable of carrying the corresponding  $\chi$ DL annotation, which is therefore a documented limitation of the target schema rather than a translation error or a disqualification of a misformatted action (we use the single term schema dropout for this category throughout the manuscript); and (iii) mistranslated, meaning that the output row disagreed with the format description or lost information that the schema could in principle have carried. In parallel, every reagent identifier, zone assignment, temperature setpoint, hold time, stir rate, mass, and volume was checked for faithful propagation through the mass-equivalent loading heuristic defined by eqn (1). The outcome is summarised in Table 5.

Aggregated over the four cases, 47  $\chi$ DL actions were processed in total, of which 44 were correctly expanded and 3 were schema dropouts, *i.e.*, intentionally omitted because the current Chemspeed schema has no corresponding field (one

EvacuateAndRefill in v91p0027 and two Purge actions in v97p0054); no action was mistranslated. Interpreted as action-level metrics against the source  $\chi$ DL, this corresponds to a correctly-expanded rate of 44/47 (93.6%) and a schema-dropout rate of 3/47 (6.4%), with the two categories together accounting for all 47 actions. The schema dropouts are not produced incorrectly by the pipeline: they are not emitted into the final execution CSV because there is currently no Chemspeed field to emit them into. The final provenance sidecar nevertheless retains these dropped rows with `status=dropped_by_downstream_converter`, which permits auditors to distinguish schema-level omissions from unobserved errors. The two of these limitations that have direct chemical relevance are addressed elsewhere in this paper: the absence of per-vial inert-gas handling (EvacuateAndRefill and Purge) is justified operationally by the sub-100 ppm deck-level inertisation of the present workstation (Sections 2.1 and 4.5), and the dropwise attribute with explicit addition time has a concrete planned implementation route through an optional `flow_rate` field (Case v97p0054 below). We regard Table 5 as a transparent step-fidelity audit that is now supported by deterministic row-level provenance files for the reference conversion route, while keeping an automated, multi-annotator evaluation against independently constructed reference CSVs as a natural next step for statistical validation.

We note three caveats for interpreting this analysis honestly. First, it is a single-annotator expert-curated analysis performed by the authors rather than a blind comparison against independently constructed reference CSVs, so it cannot substitute for inter-annotator agreement statistics. Second, because the pipeline is deterministic at the output layer (Section 3.3), the fidelity values reported in Table 5 are run-invariant for the converged `*_converted.csv` used by the workstation and are therefore not averaged over independent LLM runs. Third, the analysis covers action-level and parameter-level fidelity but not the chemical adequacy of the reaction itself, which is the domain of hardware execution and was established only for the benzylation case study. With these caveats stated, Table 5 supports the interpretation that the translation layer reported here is step-faithful within the current scope of the Chemspeed schema across all four benchmark cases, with remaining gaps

**Table 5** Expert-curated step-level fidelity analysis for the four benchmark cases, obtained by manual comparison of each .xd1 input, intermediate CSV, and converted CSV. "Correctly expanded" refers to  $\chi$ DL actions that produced the expected Chemspeed row, triplet, or cross-vessel transfer. "Schema dropout" refers to  $\chi$ DL actions or attributes intentionally omitted because the current Chemspeed format description does not expose a corresponding field; these are documented schema limitations rather than translation errors and are listed explicitly so that the optional annotations not yet captured by the pipeline are fully transparent. "Mistranslated" refers to  $\chi$ DL actions whose output row is missing or disagrees with the format description. Temperatures, hold times, stir rates, and cross-vessel transfers are listed in the units produced by the converter

| Fidelity item                                | v93p0063                                     | v91p0027                                      | v92p0328                                                    | v97p0054                                                   |
|----------------------------------------------|----------------------------------------------|-----------------------------------------------|-------------------------------------------------------------|------------------------------------------------------------|
| $\chi$ DL actions in procedure               | 11                                           | 12                                            | 10                                                          | 14                                                         |
| $\chi$ DL actions correctly expanded         | 11/11                                        | 11/12                                         | 10/10                                                       | 12/14                                                      |
| $\chi$ DL actions omitted as schema dropouts | 0                                            | 1 (EvacuateAndRefill, $\times 2$ cycles)      | 0                                                           | 2 (Purge $\times 2$ )                                      |
| Additional attribute-level schema dropouts   | None                                         | None                                          | None                                                        | dropwise="True" on one Add                                 |
| $\chi$ DL actions mistranslated              | 0                                            | 0                                             | 0                                                           | 0                                                          |
| Reagent $\rightarrow$ zone assignments       | 4/4                                          | 7/7                                           | 4/4                                                         | 4/4                                                        |
| Temperature setpoints preserved              | Reflux $\rightarrow$ 81.6 $^{\circ}\text{C}$ | 60 $^{\circ}\text{C}$ , 25 $^{\circ}\text{C}$ | 0 $^{\circ}\text{C}$ , 110 $^{\circ}\text{C}$               | −10 $^{\circ}\text{C}$ , 25 $^{\circ}\text{C}$             |
| Hold-time propagation (to min)               | 10, 20, 120                                  | 960                                           | 10, 960                                                     | 15, 10, 1, 30                                              |
| Stir-rate propagation                        | n/a                                          | n/a                                           | n/a                                                         | 200 rpm                                                    |
| Cross-vessel transfer preservation           | n/a                                          | n/a                                           | Zone-2 $\rightarrow$ zone-3; 380.9 mL, scaled full transfer | Zone-5 $\rightarrow$ zone-1; 90.4 mL, scaled full transfer |
| Global scaling applied to all masses/volumes | ✓                                            | ✓                                             | ✓                                                           | ✓                                                          |

localised entirely at the schema level rather than at the LLM mapping layer.

The three additional cases are summarised below. The corresponding execution-format CSV files are included in the source archive accompanying this manuscript and are not reproduced here in full.

## 5.2 Case v91p0027 (Sonogashira coupling<sup>23</sup>)

The  $\chi$ DL input charges 2-iodobenzamide together with three catalyst components ( $\text{PPh}_3$ ,  $\text{CuI}$ ,  $\text{Pd}(\text{OAc})_2$ ) into the reactor, followed by DMF,  $\text{Et}_3\text{N}$ , and 1-hexyne. After `StartStir` and `EvacuateAndRefill` ( $\text{N}_2$ , two cycles), the mixture is heated to 60  $^{\circ}\text{C}$  and held for 16 h. The generated CSV correctly maps the four solid charges to four `TrsG` rows and the three liquids to `TrsV` rows. The `StartStir` declaration is emitted as its own `Stir`, `zone-1`, row, and the subsequent timed `Stir` step (16 h) is then rendered as the canonical `Stir/Wait/StirO` triplet (`Stir`, `zone-1`, `Wait`, 960.0000/`StirO`, `zone-1`) in accordance with the format-description rule that requires each explicit `StartStir` and each timed `Stir` to emit its own dedicated `Stir` row. The `StopHeatChill` action is correctly translated into a final `HeatCool`, `zone-1`, 25.0000 row representing the return to ambient temperature. As described in Sections 2.1 and 4.5, the `EvacuateAndRefill` step is intentionally not emitted into the converted CSV because the Chemspeed SWING workstation used in this work performs an instrument-level inert-gas purge of the entire deck, reducing the residual oxygen concentration inside the enclosure to below 100 ppm; we have empirically confirmed that oxygen-sensitive organometallic reactions, including Pd/Cu-catalysed couplings

of the Sonogashira type considered here, proceed reliably under this atmosphere. The converter therefore maps `VacRefill` to an empty Chemspeed command in the current code path. Operators using a workstation without enclosure-level inertisation would need to supply this functionality through a different mapping rule or a hardware-side procedure; we note this as a deployment-dependent design choice rather than a translation defect.

## 5.3 Case v92p0328 (Di Maso *et al.*, Michael-addition step<sup>24</sup>)

This case is drawn from the Di Maso *Organic Syntheses* procedure for 4-methyl-1-(2-(phenylsulfonyl)ethyl)-2,6,7-trioxabicyclo [2.2.2]octane and exercises a multi-vessel workflow with an intermediate transfer between zones. The  $\chi$ DL input first charges water and maleic anhydride into a preparation vessel (`mix_r1_s1`), stirs for 10 min, transfers the resulting solution into the reactor, cools to 0  $^{\circ}\text{C}$ , charges sodium benzenesulfonate and acetic acid, starts stirring, and then heats to 110  $^{\circ}\text{C}$  for 16 h. The generated CSV correctly distinguishes the preparation zone (`zone-2`) from the reaction zone (`zone-3`). The quantity-omitted cross-zone `Transfer` is recovered by the full-transfer total-mixture rule specified in the Chemspeed format description: the LLM sums the explicit liquid `Add` volume (water, 370 mL) and the solid-contribution volume of the `Add` of maleic anhydride (10.9 g interpreted at a density of 1 g mL<sup>−1</sup>, contributing 10.9 mL) to give a transferred volume of 380.9 mL, which is emitted in the intermediate CSV as `TrsV`, "mix\_r1\_s1", reactor1, 380.9 mL, and scaled by the downstream converter to `TrsV`, `zone-2`, `zone-3`, 5.7517,. The converter cools the reaction zone to 0  $^{\circ}\text{C}$  as `HeatCool`,

zone-3, 0.0000 before the subsequent solid and liquid charges, and translates the heating step to 110 °C as Heat-Cool, zone-3, 110.0000. The 10 min preparation-vessel hold is converted to Wait, 10.0000 and the 16 h reactor hold to Wait, 960.0000, as expected.

#### 5.4 Case v97p0054 (Step A, DPPH precursor<sup>25</sup>)

This case is the most demanding of the four because it combines a sub-ambient temperature setpoint, an explicit stirring rate of 200 rpm, a dropwise addition, and a side-flask preparation that must subsequently be transferred into the main reactor. The  $\chi$ DL input charges *tert*-butyl-*N*-hydroxycarbamate and dichloromethane into the main reactor, performs a Purge with nitrogen, starts stirring at 200 rpm, cools to  $-10\text{ }^{\circ}\text{C}$ , dropwise-adds triethylamine over 10 min, then performs a second Purge on the side flask `mix_r2_r4` charges diphenylphosphinic chloride and dichloromethane into the side flask, stirs briefly, transfers the side-flask contents into the main reactor over 30 min, and finally releases both stirring and temperature control. The generated CSV preserves the `Stir, zone-1, 200.0000` command with the explicit rate, sets `HeatCool, zone-1, -10.0000`, and represents the side-flask preparation as a separate destination (`zone-5`) into which diphenylphosphinic chloride is charged *via* `TrsV, zone-4, zone-5` (the source  $\chi$ DL Add element carries both `mass="50.0 g"` and `volume="40.4 mL"`, and the LLM selected the volume form in this run) and dichloromethane is charged *via* `TrsV, zone-2, zone-5`. The quantity-omitted Transfer of the side-flask contents into `zone-1` is then recovered by the full-transfer total-mixture rule as the sum of the two liquid Add volumes ( $40.4\text{ mL} + 50\text{ mL} = 90.4\text{ mL}$ ) and is emitted in the intermediate CSV as `TrsV, "mix_r2_r4", reactor1, 90.4 mL`, which the downstream converter scales to `TrsV, zone-5, zone-1, 1.5436`, after applying the global loading factor. The terminal `StopStir` and `StopHeatChill` actions of the  $\chi$ DL are correctly emitted as `StirOff, zone-1` and a final `HeatCool, zone-1, 25.0000` row representing the return to ambient temperature, exactly as in the equivalent terminal block of case `v91p0027`. The two Purge actions are rendered into the intermediate CSV as `VacReOff` rows and are then omitted as schema dropouts by the downstream converter for the same deck-level inertisation reason described in Sections 2.1 and 4.5. The dropwise nature of the triethylamine addition and the 30 min duration of the final transfer are not currently encoded in the converted CSV because the Chemspeed format description in the present implementation does not include an explicit dropwise flag or addition-time field on `TrsV` rows; instead, the intermediate CSV emits the `TrsV` row followed by a `Wait` row whose duration matches the original addition time, which preserves the overall timing but not the instantaneous slow-addition profile. This is a known schema-level limitation rather than a translation error, and dropwise support is identified as an immediate target for future work: the natural implementation route is to extend the Chemspeed format description with an optional `Row_rate` (or addition-time) field on `TrsV` rows and to have the LLM populate

that field whenever the source `χDL Add` element carries `dropwise="True"` or an explicit addition time attribute, without any change to the Stage 3 deterministic post-processor.

## 5.5 Discussion of the benchmark

Across the four cases, the pipeline produces syntactically well-formed Chemspeed CSV files that preserve the principal additions, holds, and temperature setpoints of the source  $\chi$ DL. The step-level fidelity analysis of Table 5 aggregates to 44 correctly expanded and 3 schema dropouts out of 47  $\chi$ DL actions across the four cases, with no mistranslations observed. Three observations are worth highlighting. First, the timed-Stir expansion specified in the Chemspeed format description is followed uniformly across all four cases: every timed Stir step in the input is mapped to a canonical Stir/Wait/Stir triple, and every explicit StartStir declaration is mapped to its own dedicated Stir row rather than being absorbed into a neighboring triplet, as required by the format-description rule that forbids merging a StartStir and a subsequent time-bounded Stir into a single Stir line. For vessels that accumulate multiple stirring events (e.g. the reactor in v93p0063, which is subject to two StartStir tags and two timed Stir steps), the emitted CSV contains one Stir row per explicit stirring event, producing the corresponding count of visible Stir rows in the intermediate CSV. Second, optional  $\chi$ DL annotations such as dropwise="True", addition time, and inert-gas exchange (EvacuateAndRefill in v91p0027 and Purge in v97p0054) are omitted as schema dropouts because the Chemspeed format description does not currently provide corresponding fields, which Table 5 makes transparent on a per-case basis; capturing these annotations would require both a richer Chemspeed schema and prompt updates, and a concrete implementation route for the dropwise case is identified in the v97p0054 description above. Third, the absence of observed mistranslations at the action level across all four cases, including the most demanding v97p0054 case with sub-ambient control and side-flask preparation, indicates that under the current prompt and with low-temperature sampling the remaining gaps are localised entirely at the schema level rather than at the LLM mapping layer. Although the benchmark is intentionally limited to four procedures, Table 5 provides a per-case step-fidelity accounting that can be audited row by row against the supplied source archive and the generated provenance sidecars. A larger, automated benchmark with blind multi-annotator scoring against independently constructed reference CSVs, hardware validation of the additional cases, and a verifier-coupled LLM variant that monitors action-level alignment automatically, are the natural next steps and are identified as future work in Section 4.6. As an orthogonal cross-check, feeding the same four  $\chi$ DL inputs through the deterministic rule-based converter `xd12chemspeed_rulebased.py` introduced in Section 2.1 and then through the same `ChemspeedSystemConverter.py` post-processor yields intermediate and final `*_converted.csv` files that are byte-for-byte identical to those of the LLM pipeline in all four cases. The same route writes `<stem>_provenance.csv` and

`<stem>_converted_provenance.csv` files for each benchmark case, which confirms that the step-level fidelity reported in Table 5 is not an artefact of the LLM stage but is inherent to the format-description rules themselves and is now linked to explicit row-level evidence.

## 6 Significance for materials acceleration platforms

The present software extends our previous studies on structured representation and review into platform-level execution on a commercial materials-synthesis workstation. OSPAR provided a semantically grounded representation of synthesis actions.<sup>17</sup> The subsequent review framework improved the interpretability and correction of automatically converted procedures.<sup>18</sup> The software reported here adds an execution-oriented interface that links those structured procedures to a Chemspeed installation, which is representative of commercial automation infrastructure used in many materials acceleration platforms.<sup>5,9,10</sup>

This connection is important because structured procedural data become substantially more useful when they can be translated into concrete robotic actions on hardware that is already present in the laboratory. An  $\chi$ DL-centered representation can function as a portable intermediate layer for synthesis knowledge across the community.<sup>13,14</sup> A Chemspeed-oriented execution layer can then convert this knowledge into commands that are suitable for a commercial platform without requiring laboratories to adopt Chemputer hardware or a fully bespoke robotic stack. Even in its current proof-of-concept form, the present work therefore contributes to interoperability between literature mining, structured synthesis representation, and robotic execution in the context of materials research.

In addition, the software architecture is compatible with continued improvement of upstream natural-language-processing modules. Improvements in transformer-based action-sequence models,<sup>16</sup> LLM-based verifier-assisted generation,<sup>19</sup> and our own review-oriented framework<sup>18</sup> can be connected to the same downstream execution layer. This modularity is relevant for future closed-loop systems in which literature understanding, structured representation, and automated execution are integrated into a unified workflow.

Finally, we state explicitly the expected usability of the framework beyond organic synthesis, and the limitations to be anticipated when transferring it. The operation classes exercised in this work—gravimetric solid dispensing, volumetric liquid dispensing, stirring, temperature control, and timed holds—are generic liquid- and solid-handling primitives rather than organic-synthesis-specific operations, and the same translation layer is therefore directly applicable to workflows such as formulation and electrolyte screening, polymer and nanoparticle syntheses, and catalyst-precursor preparation on the same class of hardware. Three limitations should be expected in such transfers. First, the  $\chi$ DL action vocabulary and the available extraction corpora are strongly oriented toward

organic-synthesis prose, so upstream literature-to- $\chi$ DL conversion may require domain-specific annotation effort before the present layer can be used. Second, unit operations that are common in materials domains but absent from the current command vocabulary—filtration and phase separation (Section 4.6), calcination, thin-film deposition, ball milling, and prolonged slurry handling—require schema extensions and, in some cases, hardware beyond the present configuration. Third, the fixed-density loading heuristic of eqn (1) and the dispensing assumptions of the liquid-handling unit are stressed by highly viscous, heterogeneous, or gas-evolving media, and per-vial atmosphere control may be mandatory for moisture- or oxygen-sensitive inorganic syntheses in installations without enclosure-level inertisation (Section 3.5).

## 7 Conclusions

We developed `xd12chemspeed`, a software framework that translates literature-derived  $\chi$ DL procedures into Chemspeed-oriented execution-format programs. The framework builds upon our previous studies on OSPAR and on the review of automatically converted structured synthesis procedures,<sup>17,18</sup> and it is designed to be integrated downstream of LLM-based natural-language  $\rightarrow$   $\chi$ DL pipelines such as CLAIRify.<sup>19</sup> The use of an LLM-assisted mapping stage at the  $\chi$ DL  $\rightarrow$  Chemspeed step, rather than a purely rule-based translator, is a deliberate architectural choice that keeps the whole natural-language  $\rightarrow$   $\chi$ DL  $\rightarrow$  Chemspeed chain homogeneous in its treatment of the  $\chi$ DL action vocabulary and allows the mapping to adapt to new or rarely encountered  $\chi$ DL elements through prompt-level updates alone. The present implementation reads  $\chi$ DL files, generates intermediate Chemspeed instructions through this LLM-assisted mapping stage guided by explicit format descriptions, and converts the result into a workstation-oriented final format through selective field quotation, sequential zone renaming, reflux-temperature substitution, and heuristic global scaling. Deterministic Stage 3 post-processing absorbs the residual stochasticity of the LLM stage, as demonstrated by byte-for-byte identical final CSVs across ten independent runs with both the remote `gpt-4o-2024-05-13` model and the on-premises `gpt-oss-120b` model. A perturbation benchmark over formatting, lexical, and typographical variants of the same four  $\chi$ DL inputs further quantifies the tolerance of the LLM-assisted route to surface-level input variation and documents the fail-safe character of the observed deviations (Section 3.4).

End-to-end validation was demonstrated by loading the automatically generated program into `xd12chemspeed_executor.app` and executing it without manual modification on a Chemspeed SWING workstation at 2.65 mmol scale, affording the target product in 84.4% isolated yield (546.12 mg) with spectroscopic data consistent with the literature, comparable to the half-scale (27 mmol) literature yield of 86% (5.70 g) reported in Note 7 of the original *Organic Syntheses* article.<sup>22</sup> A complementary file-generation benchmark covering the case study and three additional *Organic Syntheses* procedures (four cases, 47  $\chi$ DL actions in total) showed that the pipeline correctly handles

multi-solid charges, sub-ambient HeatCool setpoints, explicit stirring rates, and cross-vessel transfers; a retrospective expert-curated step-level fidelity analysis across the same four cases classifies 44  $\chi$ DL actions as correctly expanded and 3 as schema dropouts, intentionally omitted from the final execution CSV because the current Chemspeed schema has no corresponding field (one EvacuateAndRefill and two Purge actions), with no action-level mistranslations observed, so that the remaining gaps are localised entirely at the schema level rather than at the LLM mapping layer. Inert-gas handling is delegated to the workstation enclosure rather than encoded per vial, which is justified by the measured sub-100 ppm oxygen level inside the deck of the present Chemspeed SWING workstation under standard nitrogen purging and is consistent with the successful execution of oxygen-sensitive organometallic reactions in our laboratory under the same conditions; this remains a deployment-dependent design choice for other installations. Extension of the Chemspeed format description to support dropwise addition and explicit addition times, on-deck filtration/solid-phase extraction and online-analysis commands, hardware validation of the three additional benchmark cases, verifier-coupled LLM source-step alignment, and blind multi-annotator scoring over a larger corpus are the next targets for future work. As an architectural cross-check, we also implemented a purely rule-based deterministic converter, `xdl2chemspeed_rulebased.py`, that performs the  $\chi$ DL  $\rightarrow$  intermediate CSV mapping by XML parsing and dictionary-based dispatch rather than by LLM inference, and we verified that feeding its output through the same downstream converter produces intermediate and final `*_converted.csv` files that are byte-for-byte identical to the LLM-based pipeline output across all four benchmark cases. The rule-based route additionally writes source-step provenance files, and the downstream converter writes final-output provenance files that retain unit conversions, zone assignments, scaling factors, software-version metadata, chemical-zone table links, file hashes, and dropped-row status records without modifying the Chemspeed execution CSV. This equivalence supports the interpretation that the Chemspeed format description is semantically complete for these procedures and clarifies that the LLM stage in the primary pipeline is retained for architectural homogeneity with upstream natural-language  $\rightarrow$   $\chi$ DL LLM pipelines and for prompt-level extensibility, rather than to compensate for any unrecoverable ambiguity at the  $\chi$ DL  $\rightarrow$  Chemspeed boundary. Within the scope demonstrated here, the framework provides a practical and inspectable bridge between machine-readable synthesis procedures and commercial laboratory automation.

## Author contributions

Y. N. contributed to conceptualisation, software revision, validation, formal analysis, investigation, writing of the original draft, and review and editing. K. M. contributed to conceptualisation, software development, data curation, validation, and review and editing. S. A. contributed to Chemspeed system implementation, hardware execution, validation, and review

and editing. M. Y. contributed to supervision, conceptualisation, project administration, and review and editing. All authors approved the manuscript and the submitted software archive.

## Conflicts of interest

There are no conflicts to declare.

## Data availability

The code and data supporting this article are openly available in the GitHub repository `yuuyanagata/xdl2chemspeed` at <https://github.com/yuuyanagata/xdl2chemspeed> and are archived on Zenodo (concept DOI <https://doi.org/10.5281/zenodo.20327326>, which always resolves to the most recent version). The release version used for this revised manuscript is `rev22-final`, archived under the version DOI <https://doi.org/10.5281/zenodo.21358276>. The archive contains `xdl2chemspeed.py`, `xdl2chemspeed_rulebased.py`, `ChemspeedSystemConverter.py`, prompt-description files, the solvent boiling-point table, the four benchmark  $\chi$ DL files, the generated intermediate CSV files, the generated converted CSV files, the intermediate and final row-level provenance sidecars, the reproducibility-run outputs, the perturbation-robustness evaluation kit (`robustness_eval/`) with the perturbed  $\chi$ DL inputs and evaluation report, the unit-conversion regression test, and the provenance regression test. The data supporting the experimental validation, including the detailed procedure and the  $^1\text{H}$  NMR spectrum of the isolated product, are included in the supplementary information (SI). Supplementary information: experimental procedure for the automated synthesis of 4-benzoyloxy-1,2-dimethoxybenzene (experiment v93p0063), including reagent quantities, the sequence of operations carried out on the Chemspeed SWING workstation, and the post-run workup and purification steps. It also summarizes the row-level provenance sidecars that accompany the benchmark outputs. The  $^1\text{H}$  NMR spectrum of the isolated product recorded at 399 MHz in  $\text{CDCl}_3$  is shown in Fig. S1. Fig. S2 presents the deck layout and zone arrangement of the Chemspeed platform used in this study. Fig. S3 shows a screenshot of `xdl2chemspeed_executar.app`, illustrating the hierarchical task tree and the parameter fields by which each command type is dispatched to the Chemspeed workstation hardware. The SI additionally documents the implementation details of the LLM-assisted route (client construction, sampling configuration, token-limit handling, field-quoting post-processing, and CSV-arity conventions). It also describes the perturbation-robustness protocol and the corresponding evaluation scripts. See DOI: <https://doi.org/10.1039/d6dd00301j>.

## Acknowledgements

This work was supported by JSPS KAKENHI (grant numbers JP23H03810 and JP23K18500) and by the JST Support for Pioneer Research Initiated by the Next Generation (SPRING)

program (grant number JPMJSP2119). Part of this work was additionally supported by JST ERATO (grant number JPMJER1903) and by the Institute for Chemical Reaction Design and Discovery (ICReDD) at Hokkaido University, established under the World Premier International Research Center Initiative (WPI) of the Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan. A part of this work was further supported by the New Energy and Industrial Technology Development Organization (NEDO) Project to Develop Common Platforms Supporting the Widespread Use of Hydrogen. The authors acknowledge the foundations established by our previous studies on OSPAR and structured procedure review, which enabled the present integration work. During manuscript revision, AI-assisted editorial and code-review tools were used for language polishing, consistency checks, and identification of software defects. The authors reviewed all revised text, code, data, and conclusions and accept responsibility for the final content.

## References

- 1 F. Häse, L. M. Roch and A. Aspuru-Guzik, *Trends Chem.*, 2019, **1**, 282–291.
- 2 E. Stach, B. DeCost, A. G. Kusne, J. Hattrick-Simpers, K. A. Brown, K. G. Reyes, J. Schrier, S. J. L. Billinge, T. Buonassisi, I. Foster, C. P. Gomes, J. M. Gregoire, A. Mehta, J. Montoya, E. Olivetti, C. Park, E. Rotenberg, S. K. Saikin, S. Smullin, V. Stanev and B. Maruyama, *Matter*, 2021, **4**, 2702–2726.
- 3 M. Seifrid, R. Pollice, A. Aguilar-Granda, Z. Morgan Chan, K. Hotta, C. T. Ser, J. Vestfrid, T. C. Wu and A. Aspuru-Guzik, *Acc. Chem. Res.*, 2022, **55**, 2454–2466.
- 4 M. Abolhasani and E. Kumacheva, *Nat. Synth.*, 2023, **2**, 483–492.
- 5 G. Tom, S. P. Schmid, S. G. Baird, Y. Cao, K. Darvish, H. Hao, S. Lo, S. Pablo-García, E. M. Rajaonson, M. Skreta, N. Yoshikawa, S. Corapi, G. D. Akkoc, F. Strieth-Kalthoff, M. Seifrid and A. Aspuru-Guzik, *Chem. Rev.*, 2024, **124**, 9633–9732.
- 6 C. Bayley, *et al.*, *Matter*, 2024, **7**, 2382–2398.
- 7 S. Lo, S. G. Baird, J. Schrier, B. Blaiszik, N. Carson, I. Foster, A. Aguilar-Granda, S. V. Kalinin, B. Maruyama, M. Politi, H. Tran, T. D. Sparks and A. Aspuru-Guzik, *Digital Discovery*, 2024, **3**, 842–868.
- 8 A. V. Tobias and A. Wahab, *R. Soc. Open Sci.*, 2025, **12**, 250646.
- 9 B. P. MacLeod, F. G. L. Parlane, C. C. Rupnow, K. E. Dettelbach, M. S. Elliott, T. D. Morrissey, T. H. Haley, D. J. Dvorak, M. B. Rooney, J. R. Deeth, V. Lai, G. J. Ng, H.-W. Lee, S. Rocha, M.-A. Légaré, A. Balasubramaniam, F. Denes, G. Tom, D. Farano, H. Molero, M. R. Jones, S. Noormohammad, M. A. G. Koffas and C. P. Berlinguette, *Sci. Adv.*, 2020, **6**, eaaz8867.
- 10 B. Burger, P. M. Maffettone, V. V. Gusev, C. M. Aitchison, Y. Bai, X. Wang, X. Li, B. M. Alston, B. Li, R. Clowes, N. Rankin, B. Harris, R. S. Sprick and A. I. Cooper, *Nature*, 2020, **583**, 237–241.
- 11 L. M. Roch, F. Häse, C. Kreisbeck, T. Tamayo-Mendoza, L. P. E. Yunker, J. E. Hein and A. Aspuru-Guzik, *Sci. Robot.*, 2018, **3**, eaat5559.
- 12 S. Steiner, J. Wolf, S. Glatzel, A. Andreou, J. M. Granda, G. Keenan, T. Hinkley, G. Aragon-Camarasa, P. J. Kitson, D. Angelone and L. Cronin, *Science*, 2019, **363**, eaav2211.
- 13 S. H. M. Mehr, M. Craven, A. I. Leonov, G. Keenan and L. Cronin, *Science*, 2020, **370**, 101–108.
- 14 R. Rauschen, M. Guy, J. E. Hein and L. Cronin, *Nat. Synth.*, 2024, **3**, 488–496.
- 15 M. Šiaučiulis, C. Knittl-Frank, S. H. M. Mehr, E. Clarke and L. Cronin, *Nat. Commun.*, 2024, **15**, 10261.
- 16 A. C. Vaucher, P. Schwaller, J. Geluykens, V. H. Nair, A. Iuliano and T. Laino, *Nat. Commun.*, 2020, **11**, 3601.
- 17 K. Machi, S. Akiyama, Y. Nagata and M. Yoshioka, *J. Chem. Inf. Model.*, 2023, **63**, 6619–6628.
- 18 K. Machi, S. Akiyama, Y. Nagata and M. Yoshioka, *Digital Discovery*, 2025, **4**, 172–180.
- 19 N. Yoshikawa, M. Skreta, K. Darvish, S. Arellano-Rubach, Z. Ji, L. B. Kristensen, A. Z. Li, Y. Zhao, H. Xu, A. Kuramshin, *et al.*, *Auton. Robots*, 2023, **47**, 1057–1086.
- 20 M. Seifrid, F. Strieth-Kalthoff, M. Haddadnia, T.-C. Wu, E. Alca, L. Bodo, S. Arellano-Rubach, N. Yoshikawa, M. Skreta, R. Keunen and A. Aspuru-Guzik, *Digital Discovery*, 2024, **3**, 1319–1326.
- 21 S. Kim, *et al.*, *Nucleic Acids Res.*, 2025, **53**, D1516–D1525.
- 22 S. Okaya, K. Okuyama, K. Okano and H. Tokuyama, *Org. Synth.*, 2016, **93**, 63–74.
- 23 N. Okamoto, K. Takeda and R. Yanada, *Org. Synth.*, 2014, **91**, 27–38.
- 24 M. J. Di Maso, M. A. S. Peter and J. T. Shaw, *Org. Synth.*, 2015, **92**, 328–341.
- 25 T. Benkovics, A. J. Neel, R. Zhao and G. J. Hughes, *Org. Synth.*, 2020, **97**, 54–65.