A universal system for digitization and automatic execution of the chemical synthesis literature

S. Hessam M. Mehr*, Matthew Craven*, Artem I. Leonov*, Graham Keenan, Leroy Cronin†

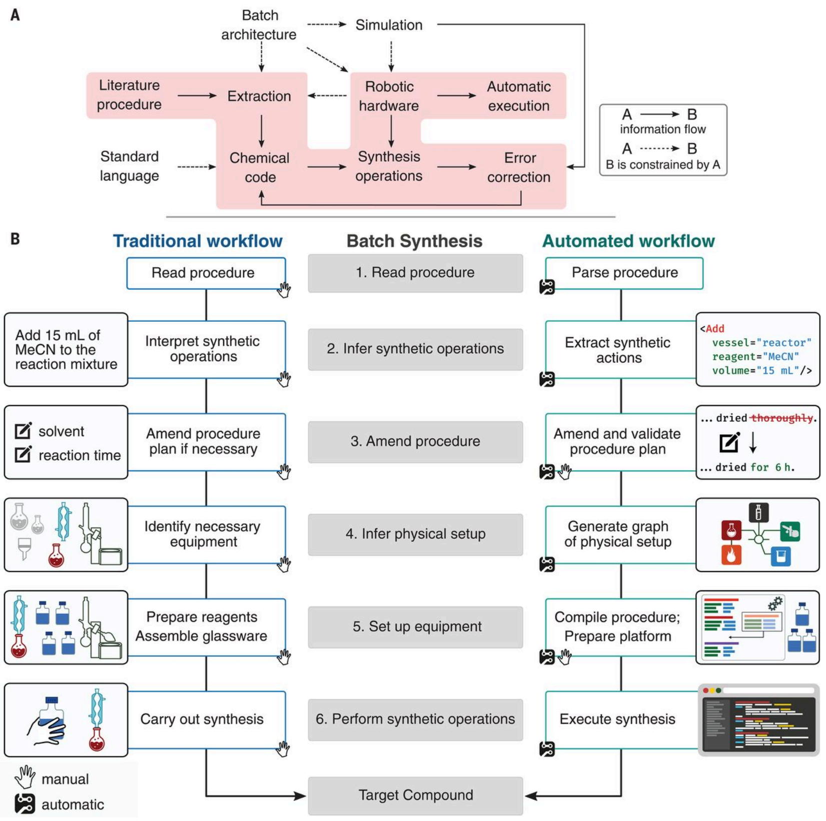
School of Chemistry, University of Glasgow, Glasgow G12 8QQ, UK.

†Corresponding author. Email: lee.cronin@glasgow.ac.uk
* These authors contributed equally to this work.

Robotic systems for chemical synthesis are growing in popularity but can be difficult to run and maintain because of the lack of a standard operating system or capacity for direct access to the literature through natural language processing. Here we show an extendable chemical execution architecture that can be populated by automatically reading the literature, leading to a universal autonomous workflow. The robotic synthesis code can be corrected in natural language without any programming knowledge and, because of the standard, is hardware independent. This chemical code can then be combined with a graph describing the hardware modules and compiled into platform-specific, low-level robotic instructions for execution. We showcase automated syntheses of 12 compounds from the literature, including the analgesic lidocaine, the Dess-Martin periodinane oxidation reagent, and the fluorinating agent AlkylFluor.

Synthetic chemistry remains labor intensive, and some protocols suffer from errors or ambiguity (1, 2). Recently, there has been rapid growth in the development of robotic synthesis of molecules (3–5), but new developments are limited to specific reaction types, and a universal approach for the automatic encoding and validation of the chemical synthesis literature is lacking, which means that automation currently just displaces effort from manual labor to programming (6). The burden is further increased by the plethora of robotic solutions, which lack a common standard architecture. What is needed is an abstraction that can not only implement the literature (7) but also adapt to new synthetic methods (8–11), in accordance with a standard that ensures interoperability between hardware systems. Currently, the standard of the recording and subsequent reporting of the synthesis of new chemical compounds varies greatly, and procedures are often available only as incomplete and ambiguous prose, relying on the expert to fill in any gaps (12). This means that the quality of the data stored in many reaction databases is highly variable (13), posing many problems for reproducibility (14), as well as preventing the development of reliable digital methods for prediction of reactivity (15), new structures (16), and functions (17). These limitations have also prevented the practical digitization of chemistry (6)—i.e., the development of automated systems that could run reactions and make molecules—because of the lack of standards linking the reaction dependencies to a standard hardware control and specification, as well as a machine-readable standard for recording synthetic procedures.

A key factor hindering the digitization of chemistry is the lack of a universal chemical programming language despite the recent proliferation of chemical automation platforms. For example, we have recently developed the Chemputer (18), a programmable modular system with hardware capable of performing the fundamental processes of chemical synthesis. The Chemputer was able to automate batch synthetic procedures but was limited to executing a set of specialized low-level hardware instructions, and no uniform development environment or universal hardware interface or specification was provided to allow the code to run on other systems. Consequently, previously automated syntheses involved laborious and error-prone manual translation of the synthesis procedures to these low-level instructions, which precludes portability to other platforms and requires programming expertise as well as detailed knowledge of the system's robotic operations. Such implementations demonstrate the capabilities of the hardware but are not a suitable or sustainable way of automating chemical synthesis. Far from being restricted to the Chemputer, the absence of a universal chemical programming language threatens to undermine the feasibility of the nascent automated synthesis ecosystem. Automated platforms from different companies or research groups all have bespoke instruction sets with no obvious semantic link among them or to the literature. This broken link has prevented the digitization of chemistry: Our vast repertoire of synthetic knowledge cannot be directly executed by robots today. To address this, we envisaged that a new architecture must rely on hardware-independent instructions represented in a standard chemical language that can express the synthesis of virtually any molecule (Fig. 1A).



```
<Add
  vessel="reactor"
  reagent="MeCN"
  volume="15 mL"/>
```

... dried ~~thoroughly~~.

... dried for 6 h.

Fig. 1 Conceptual representation of chemical synthesis execution.
(A) Outline of the universal system for the automatic execution of chemical synthesis from the literature. A method of extracting the procedure is needed, followed by an algorithmic process for producing the code that conforms to a standard hardware and software architecture. Manual error correction by the expert chemist and simulated execution ensure reliability and safety. The execution route is highlighted. (B) Abstraction of batch synthesis and two possible workflows for its execution. In the traditional workflow (left), all operations specified in the middle column are carried out manually by the chemist. Most operations are automated in the digital workflow (right), although the chemist can inspect and override the system's output without having to modify the software or learn a programming language.

Whether carried out manually by a chemist or automatically by a robot, the execution of batch synthesis procedures follows a fixed set of steps, with parallels between the manual actions in the traditional workflow and their automated counterparts in the digital workflow (Fig. 1B).

Comparing the two workflows reveals the steps necessary for realizing the digital workflow. These steps all operate on or refer to an intermediate representation of the actions in the procedure. To use a sufficiently capable batch synthesis robot for the universal digitization of chemistry, we thus hypothesized that the robot's hardware abstraction must be connected to practical synthesis by an executable hardware-independent chemical programming language. This universal chemistry-oriented domain-specific markup language allows us to map the commonly reported methodological instructions to hardware-agnostic chemical instructions. Additionally, the semantic connection between this language and laboratory batch synthesis enables robotic execution of laboratory synthetic procedures without manual adaptation or modification. Instructions can thus be translated between the two without loss of information, as we no longer need to cross the large semantic gap between chemistry and robotic operations.

Herein, we present a system for the digitization of chemistry by making the chemical literature directly executable on automated synthesis platforms such as the Chemputer. To implement the system, we required a language with which synthetic procedures can be expressed succinctly by using vocabulary similar to that in the literature and executed on any compatible robotic platform. To this end, we have devised a chemical programming language and associated visual development environment that allows users to encode procedures without ambiguity and exchange them using a standard format. Our Chemical Description Language, XDL (the X pronounced "kai" in reference to "χημία," the Greek word for chemistry), achieves this goal by representing syntheses as sequences of processes taking place in abstract vessels with abstract hardware and is based on the ubiquitous XML format (19). Our chemical development environment then inspects this representation and determines which hardware components are necessary to execute it on a virtual machine, producing specifications for a platform capable of running the procedure. Given a robotic platform with the required hardware modules, which conform to the standard architecture, the generic XDL description of the procedure is compiled into an executable specific to the platform and run on the target hardware to carry out the automated synthesis (Fig. 2).

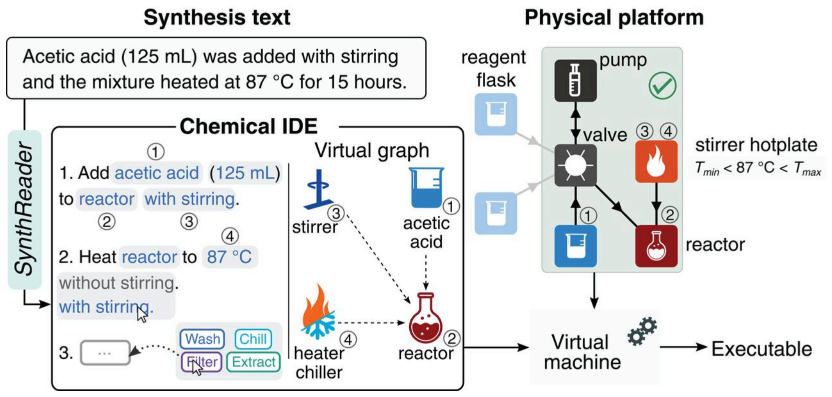


Fig. 2 System workflow.

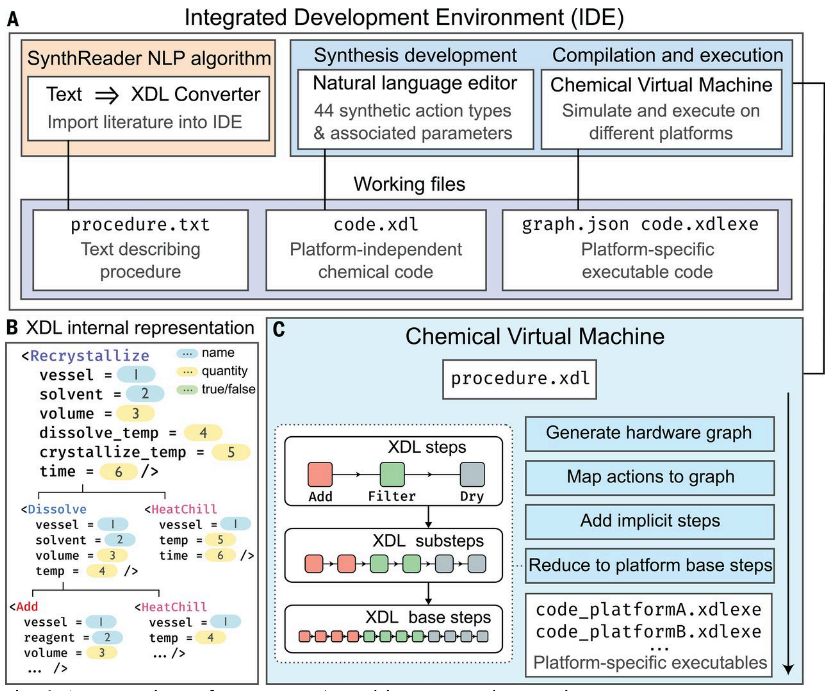
SynthReader converts synthetic procedures described using natural language into hardware-independent chemical code (XDL) within ChemIDE. The digitized procedure is represented and can be visually edited as natural language but internally stored as XML. Using the software, the user can amend the procedure or fix any potential translation errors. Once the correct chemical code has been produced, the virtual hardware used in the procedure is mapped to the user's physical platform. Compilation then combines the code with the mapped graph in a virtual machine to produce a hardware-specific executable suitable for immediate execution on the target platform. Tmax, maximum temperature; Tmin, minimum temperature.

The synthesis procedure, once written using our standard chemical programming language, provides a universal and hardware-independent way of digitizing chemical synthesis. However, there must also be a way of easily transferring syntheses written in natural language into code without programming knowledge or duplication of effort, while leveraging the expertise of the synthetic chemist. To do this, our system includes a chemical integrated development environment (ChemIDE) that facilitates importing literature procedures using a natural language processing (NLP) algorithm called SynthReader. In this context, other groups have recently applied NLP-based text mining to unstructured data from chemical synthesis texts to extract synthesis actions for both organic and inorganic reactions. This has been demonstrated by using pattern-matching techniques as well as machine learning (20–22).

Although these approaches are useful for mining vast literature datasets, we needed a system that could output a machine-readable representation of a procedure with sufficient process details to unambiguously execute the procedure on an automated platform. This goes beyond simply tagging chemical entities found in literature procedures, as it also requires a structured declaration of the location of different reagents throughout the procedure, inference of implicit process details such as separation of phases and reflux temperatures, and a development environment in which the expert chemist can unambiguously edit the output. SynthReader achieves this goal by tagging text with relevant entities, converting the tagged text to a list of actions, and then adding implicit process information and concrete reagent locations and outputting the procedure in the XDL format, which contains all of the necessary information to unambiguously execute the procedure on an automated platform. We have demonstrated the efficacy of this approach experimentally by converting literature syntheses to XDL using SynthReader and synthesizing the target molecules by executing the produced XDL.

Design and implementation of the system

The key observation underlying our system is that any synthetic step is composed of a connected series of processes (add, heat, filter, etc.). Building on this observation, our system integrates the following components to realize automated synthesis from the literature: (i) a markup language capable of representing these extracted chemical processes and combining them in a context in which they can be executed as a chemical synthesis; (ii) an IDE allowing nonprogrammers to easily edit this representation of a chemical synthesis; (iii) a tool to automatically import existing procedures into the IDE, directly from the literature; and (iv) a virtual machine capable of transforming these chemical processes into basic operations that can be directly executed on a given automated platform. The integration of these parts is shown in Fig. 3. Below, we describe each component.


A
Integrated Development Environment (IDE)
SynthReader NLP algorithm
Text ⇒ XDL Converter
Import literature into IDE
Synthesis development
Natural language editor
44 synthetic action types & associated parameters
Compilation and execution
Chemical Virtual Machine
Simulate and execute on different platforms
Working files
procedure.txt
Text describing procedure
code.xdl
Platform-independent chemical code
graph.json code.xdlexe
Platform-specific executable code
B XDL internal representation
name
quantity
true/false
C
Chemical Virtual Machine
procedure.xdl
XDL steps
Add
Filter
Dry
XDL substeps
XDL base steps
Generate hardware graph
Map actions to graph
Add implicit steps
Reduce to platform base steps
code_platformA.xdlexe
code_platformB.xdlexe
...
Platform-specific executables

```
<Recrystallize
  vessel = 1
  solvent = 2
  volume = 3
  dissolve_temp = 4
  crystallize_temp = 5
  time = 6 />

<Dissolve            <HeatChill
  vessel = 1           vessel = 1
  solvent = 2          temp = 5
  volume = 3           time = 6 />
  temp = 4 />

<Add                 <HeatChill
  vessel = 1           vessel = 1
  reagent = 2          temp = 4
  volume = 3           ... />
  ... />
```

Fig. 3 An overview of our system's architecture and operation.
(A) ChemIDE provides a visual user interface in which chemical code can be edited by using natural language. Existing literature procedures can be imported into this environment by using the SynthReader NLP algorithm, then inspected and enhanced by the user before being compiled to execute on a specific hardware target by the virtual machine. (B) XDL's internal XML-based representation propagates process information from steps to substeps. (C) A multi-step process within the virtual machine maps chemical synthesis steps to the relevant hardware modules within the target platform and recursively expands each step until it is reduced to basic hardware operations understood by the hardware.

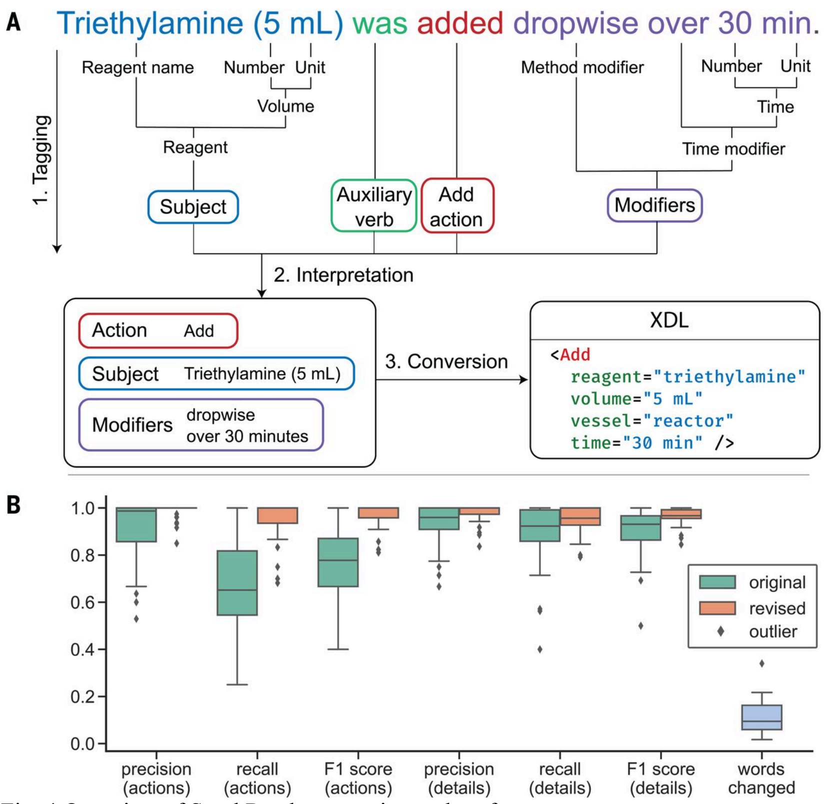
The XDL markup language was created to describe chemical synthesis in a robust, machine-readable way. The representation of a chemical synthesis as a sequence of discrete operations in XDL is the bridge between SynthReader, ChemIDE, and the virtual machine and the physical hardware operations necessary to perform that synthesis. The core of the XDL language is the XDL step. Each high-level step has a name and associated properties, and these steps and properties define the standard by which chemical syntheses can be described. Examples of top-level steps implemented in XDL are "separate," "evaporate," and "add," and a total of 44 such top-level steps have been implemented so far. These steps can be combined in a linear sequence, or branched sequences can be created and executed concurrently by using the asynchronous capabilities of XDL.

A disparate list of actions alone would hardly be sufficient for automated execution. XDL is notable for also providing the necessary experimental context, a stateful model of hardware and associated chemicals at every point in time. We designed XDL to capture information about synthetic procedures at multiple levels of abstraction, thereby allowing processes to be specified at a high level directly comparable to published methods sections but executed through low-level hardware instructions. From this representation, a graph specification of the physical platform required can be inferred. This is achieved by producing a set of hardware requirements based on the procedure, such as required hardware modules and their connectivity as well as any reagents and temperatures involved. In this respect, XDL is distinct from common chemical data interchange formats such as CML (Chemical Markup Language) in that it provides a complete executable abstraction of chemical synthesis.

As most synthetic chemists are not experienced in programming, we created ChemIDE to facilitate user-friendly editing of XDL procedures. Similar to IDEs used by software developers, this environment helps the chemist adjust the procedure, add new operations to the procedure, and use the full capabilities of the software presented here within a graphical user interface. ChemIDE allows the user to program chemical synthesis without any coding knowledge by representing each step in natural language. This means that each step is shown to the user as a simple English sentence, highlighting words and phrases that they can edit, with input options and validation built in. For example, a dry step might show “Dry contents of **reactor** for **1 h**,” in which “**reactor**” and “**1 h**” are editable, allowing the user to edit the vessel being dried and drying time. Changes made to these editable sentences are concretely linked to the underlying XDL representation, which can also be viewed in ChemIDE. Thus, users with no programming experience can interactively resolve ambiguities in the original text or amend any missing or implicit steps or process variables (movie S1).

The SynthReader NLP algorithm can extract sequences of processes from synthesis descriptions and represent them in XDL. Many recent advances in NLP have relied on the use of machine learning with large datasets of labeled text (23). However, to the best of our knowledge, there is no large dataset of labeled synthetic protocols in which the labels correspond to the final list of actions and details required for this application; therefore, we decided to build SynthReader as a domain-specific algorithm with predefined rules. This is a viable design thanks to the regular, almost mechanical language in which synthetic protocols tend to be written (24).

The text-to-XDL algorithm used in SynthReader was structured around three phases: (i) tagging [using a similar process to that used in previously published work, such as the ChemicalTagger (22)], (ii) interpretation, and (iii) conversion (Fig. 4A). In the tagging phase, different parts of the text are assigned labels, such as reagent names, volumes, or temperatures. This is achieved by using pattern-matching techniques. For example, one of the patterns used for finding solutions is “a solution of **Reagent** in **Reagent**” in which “**Reagent**” represents a phrase previously tagged as a reagent. An example phrase that would be matched by this pattern is “a solution of Oxone (181.0 g, 0.29 mol, 1.3 equiv) in deionized water (650 mL).” SynthReader contains a total of 16,582 patterns, some hard-coded, most programmatically generated from smaller hard-coded fragments.



```
<Add
  reagent="triethylamine"
  volume="5 mL"
  vessel="reactor"
  time="30 min" />
```

Fig. 4 Overview of SynthReader operation and performance.
(A) Overview of the process by which text is converted to XDL. First, the text is hierarchically tagged by pattern matching. Pattern matching is then used again to extract all actions from the labeled text with their accompanying subjects and modifiers. Finally, the extracted actions are converted to XDL. The example text here contains only one action, but the system can handle multiple actions in one sentence. (B) Accuracy statistics for the latest version of SynthReader measured against 42 literature procedures before and after making revisions to the text. The final column shows the fraction of words in the procedure text that were modified in the process of editing. The lower and upper edge of each box represent the first and third quartile of values. The lower and upper horizontal lines (whiskers) represent the lowest and highest data points within 1.5 times the interquartile range of the box edges. Outliers (data points outside this range) are represented by diamonds. For benchmarking details, see supplementary materials section 2.5.

Users are free to add, edit, and remove steps in the procedure as they wish; however, if trying to run a procedure from the literature, it is cumbersome to add every step individually. To conveniently import a literature procedure for further editing, we designed an NLP algorithm that automatically converts synthesis text to XDL. One component of the tagging process in which simple pattern matching is insufficient is the tagging of reagent names, as compiling an exhaustive list of these is not feasible. To tag reagent names, we use pattern matching with a database of common reagent names and then use a naïve Bayes classifier to determine whether candidate phrases are reagent names, excluding some specific phrases. The classifier uses probabilities that certain two-, three-, and four-letter word fragments are contained in a reagent name, calculated by using reagent names from the Reaxys database and non–reagent-name text from the Brown English language corpus. In the interpretation phase, pattern matching is again used to find common sentence structures in the tagged text and extract actions described by these sentences, along with subjects of the actions and any action-modifying phrases, resulting in a list of actions with associated information. The final conversion stage takes the list of actions produced by the interpretation stage, standardizes the details associated with each action, and converts every action to one or more XDL steps, producing a final XDL file. XDL can track the movement of reagents throughout the procedure and SynthReader can combine this information with a built-in table of physical properties to deduce process variables such as reflux temperatures and separation phases automatically. The entire text-to-XDL process is visualized for an example sentence in Fig. 4A; however, the same process is extendable to multiple paragraphs of text. SynthReader was designed with the goal of providing an accurate translation of any given synthetic protocol. However, we acknowledge that because of the flexibility of natural language, there will always be cases where the algorithm fails and information is lost, and thus, the output of any such algorithm cannot be blindly trusted. ChemIDE makes it trivial to spot and amend any such omissions or errors. Figure 4B shows the results of benchmarks measuring SynthReader's ability to recover synthetic actions and associated details from literature procedures.

Once users have imported a procedure into ChemIDE using SynthReader and edited it to their satisfaction, the resulting chemical code can be compiled and executed on various hardware platforms. This functionality, enabled by a chemical virtual machine, combines the hardware-independent XDL representation with the graph description of the target hardware and automatically associates each step with the hardware modules required for execution.

This means that one XDL file, without alteration, can be compiled and executed on multiple different platforms as long as the hardware can execute the necessary base steps. Otherwise, any missing hardware modules or modules that do not meet the required specifications are reported.

The hardware graph can be automatically generated from a template graph of a specific platform or, alternatively, from the built-in default template. The hardware requirements are checked for compatibility with the template. If they are compatible, the template graph is altered to produce a procedure graph capable of executing the procedure. If they are incompatible, the user has the option to update either the template graph or the procedure. To produce platform-specific code, the virtual machine expands each XDL step in the procedure into a tree data structure, as shown in Fig. 3C. The leaves of this tree are hardware-specific unit operations, which are directly executable on the robot. Every step effectively has a decision tree that decides, on the basis of the parameters, which lower-level steps should be executed to achieve the specified behavior. Details (e.g., vessel names, process variables) assigned to a top-level step such as "Recrystallize" are propagated to those of the appropriate substeps (heat, stir, cool). For example, recrystallization time (variable 6, Fig. 3B) is correctly attributed to the cool-down period after the dissolution of a solid.

In addition to reducing generic synthetic steps to hardware-specific unit operations, the virtual machine is responsible for supplying the necessary implicit steps. These are operations that are specific to the targeted execution platform rather than intrinsic to the chemical process. In the case of the Chemputer, for instance, residual chemicals must be removed from the liquid backbone by using an appropriate cleaning solvent regularly to prevent cross-contamination. The placement of these cleaning steps and the type of solvent used for cleaning is algorithmically determined from the sequence of operations in the synthetic protocol and a series of chemical compatibility rules. The automatic addition of these implicit steps makes digitizing chemical synthesis much simpler as the user only needs to think about the synthesis as they would on the bench, not the operation of the platform. At the moment, the high-level steps included with XDL describe common synthetic operations such as "Evaporate" and "Separate."

In the future, XDL steps can be devised to express higher-level concepts. For instance, we envision a library of XDL steps to represent name reactions, such as "SuzukiCoupling" and "DessMartinOxidation." XDL also provides support for asynchronously executed steps and steps that execute dynamically on the basis of live feedback from analytical devices. These capabilities mean that branched syntheses and more advanced laboratory techniques such as adding until a color change occurs or maintaining a certain temperature during an addition will be possible by using the XDL framework.

As the most mature hardware target available to us, the Chemputer was used to execute the syntheses in the experimental section below. However, because the chemical code is written against a chemical virtual machine rather than any specific hardware, our system can use any hardware platform with a batch synthesis architecture and an open application programming interface as a first-class target, as demonstrated by the port to a second platform (supplementary materials). The ability to target diverse robotic systems is a boon to the viability of these proliferating platforms, as it ensures that digitized synthetic knowledge is not tied to specific hardware. Furthermore, as these platforms mature and add support for more hardware, they become capable of running a larger subset of published chemical syntheses. We simulate this scenario in Fig. 5 by using SynthReader to parse and analyze the hardware requirements of 523 literature procedures. For instance, ~60% of the procedures surveyed can be executed by using the most basic six modules: the addition of the low-temperature module raises this figure by 30%. The universality of our paradigm thus extends beyond currently available hardware.

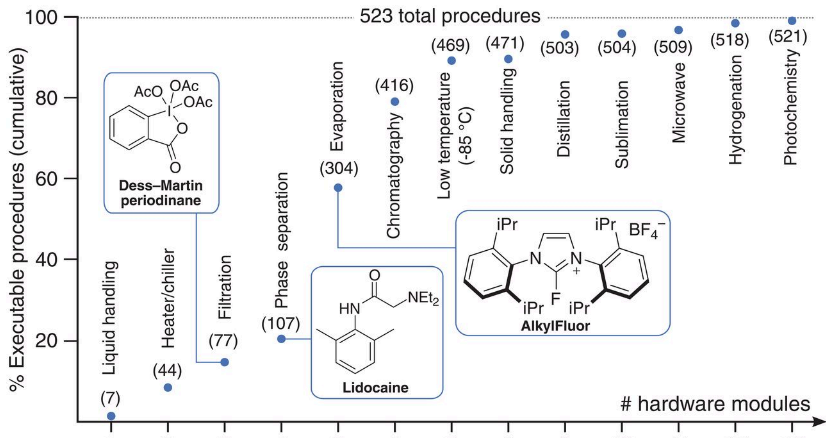


Fig. 5 Automatic analysis of the hardware requirements of literature procedures using SynthReader.
The dataset used consisted of every procedure from Organic Syntheses, volume 77 onward (559 procedures), that was analyzed by SynthReader without a fatal error (523 procedures) (9, 38). The cumulative number of executable procedures with the successive addition of each hardware module is shown in parentheses. OAc, acetate; Et, ethyl; iPr, isopropyl.

Experimental validation of the system

We have used our approach to automatically execute 12 literature procedures on the Chemputer without any additional programming or hardware changes (25). To exemplify the process, we will detail the synthesis of three compounds here: lidocaine, the Dess-Martin periodinane (DMP), and AlkylFluor. Lidocaine is used as a local anesthetic and to treat arrhythmia and epilepsy (26). The literature procedure that we consulted for the synthesis of lidocaine (27) describes a simple two-step process involving the formation of an α-chloroamide intermediate and its subsequent nucleophilic substitution reaction with diethylamine. These steps map in a straightforward fashion to the process diagram illustrated in Fig. 6A. We fed the unmodified procedure for the synthesis of lidocaine to our system to run on the Chemputer.

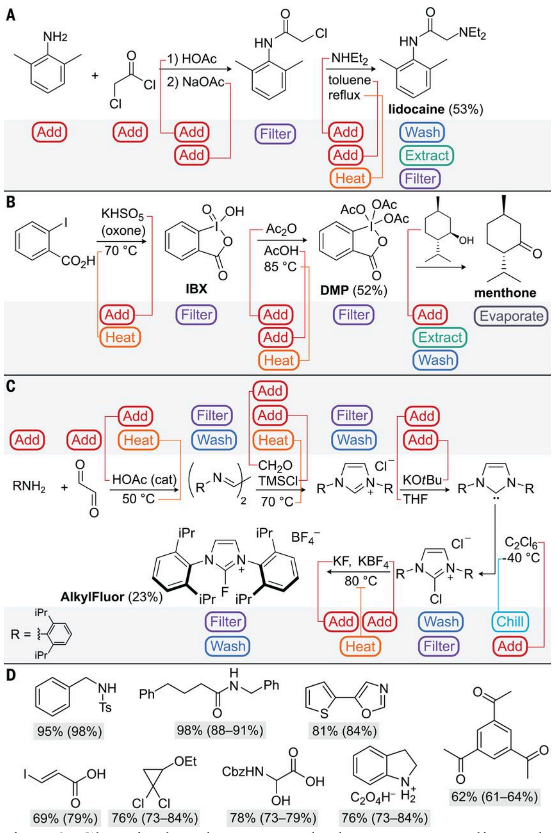


Fig. 6 Chemical schemes and the corresponding abstract chemical processes for Chemputer-implemented syntheses.
(A) Lidocaine. (B) Menthone. (C) AlkylFluor. Published synthetic procedures in the literature make these abstract processes concrete by providing a textual representation and supplying experimental details. (D) Other molecules automatically synthesized from literature procedures by using our system (literature yields are shown in parentheses for reference). See supplementary materials section 6 for details. Cbz, benzyloxycarbonyl; tBu, tert-butyl; THF, tetrahydrofuran; TMS, trimethylsilyl; Ts, toluenesulfonyl; Ph, phenyl.

On the basis of the procedure described by the XDL file, the Chemputer operated the backbone pumps and valves to automatically transfer acetic acid solvent to the jacketed filter module—which the system had identified as a suitable reactor—followed by 2,6-dimethylaniline, chloroacetyl chloride, and saturated aqueous sodium acetate. During the process, our system correctly found points at which two chemicals are mixed and controlled stirring appropriately to ensure proper mixing. On the basis of the XDL instructions, the Chemputer then performed a filtration and routed the filtrate into a waste container. The next step was executed similarly by adding diethylamine and toluene solvent, heating the jacketed filter up to reflux by using a circulation chiller to effect the substitution reaction, and using the liquid-liquid separation module to perform an acidic extraction with an aqueous hydrochloric acid solution. The detection of the liquid-liquid phase boundary is facilitated by a conductivity sensor exploiting the high conductivity of the aqueous phase compared with the organic phase (18). Finally, lidocaine was precipitated from the aqueous solution in the jacketed filter by addition of sodium hydroxide solution, filtered, and dried in the jacketed filter under vacuum. Automated execution of the literature procedure in this manner produced lidocaine in 53% yield. The chemist is responsible for ensuring that the experimental setup matches the hardware graph. For the Chemputer, this preparation step involves connecting reagent bottles for each reagent, solvent, and piece of glassware to the correct position on the liquid backbone.

The second example, the DMP, is a versatile oxidation reagent that is prized for its specificity and functional group tolerance, despite its relatively high price and moisture sensitivity. Both the preparation and use of this reagent, as well as its precursor, 2-iodoxybenzoic acid (IBX), have been the subject of recent reproducibility debates (28–32). The synthesis of DMP is a prime candidate for automation, as it is often prepared fresh, a process that is time-consuming and bears a non-negligible risk of explosion due to impurities. We applied SynthReader to three separate literature procedures: (i) a modern synthesis of IBX by oxidation of 2-iodobenzoic acid with aqueous oxone (potassium peroxymonosulfate) (33); (ii) acetylation of IBX by using acetic anhydride to form DMP (29); (iii) oxidation of menthol to menthone by using DMP to determine its activity (34). The resulting XDL files, representing the entire process, were joined together and executed on a Chemputer, giving DMP product (in 52% overall yield), which subsequently showed quantitative oxidizing activity when reacted with excess menthol (Fig. 6B) (35).

To demonstrate that our text-to-molecule machinery is not limited to short syntheses, we also converted text describing the five-step synthesis of the fluorinating agent AlkylFluor to XDL and executed it on a Chemputer platform, obtaining AlkylFluor in 23% overall yield (75% average stepwise yield) (Fig. 6C) (36, 37). Though the proliferation of automated chemical synthesis systems holds much promise, differences in the instruction set provided by the various platforms make it impractical to write portable chemical code. The approach described here is hardware-universal, meaning the software can execute a given synthetic procedure on any hardware platform as long as the platform provides the hardware instructions necessary to express the processes described in the procedure. To demonstrate this, we successfully executed the literature synthesis of the polyoxometalate $(C2H8N)8Na3[W19Mn2O61Cl(SeO3)2(H2O)2]Cl2\cdot 6H2O$ on a bespoke high-throughput chemistry robot used in our group that relies on a completely different instruction set to the Chemputer. Because this robot lacks hardware modules for heating and filtering, only a subset of the procedures executable on the Chemputer will run on it. Any unsupported actions are automatically flagged by our system when encountered.

In summary, we present an important step toward the goal of automating all aspects of synthetic chemistry—from text to molecule—with the introduction of an abstraction that allows the digitization of chemical synthesis. Although it is not yet possible to convert all the literature with our system without some manual intervention, ChemIDE allows the user to correct errors by easily inspecting the original text and confirming translation into the process steps. In the future, we will automate this verification step using a chemical autocorrect function. The NLP capabilities of SynthReader are comparable to the current state of the art (supplementary materials section 2.6) and can be easily augmented with new rules, as the design is deterministic. Real-time feedback from analytical instruments can be used to confirm that processes proceed as described by the XDL, making the system adaptive and fault tolerant. In addition, it is possible to interface the IDE with other NLP engines or hardware compilation targets, further increasing the possibilities to interface our system with any chemical robot and the broader literature.

Acknowledgements

We thank D. Angelone and S. Zalesskiy for contributing to Chemputer hardware development, A. Hammer for fruitful synthetic discussions, D. Salley for help with implementing XDL on Chemobot, and P. J. Kitson for contributing ideas and helping with manuscript preparation. We also thank M. Symes, N. Bell, S. Rohrbach, R. Hartley, and L. Wilbraham for comments on the manuscript. **Funding:** We thank the following funders: EPSRC (grant nos. EP/H024107/1, EP/J00135X/1, EP/J015156/1, EP/K021966/1, EP/K023004/1, EP/L023652/1) and ERC (project 670467 SMART-POM). This research was developed with funding from the Defense Advanced Research Projects Agency (DARPA). The views, opinions, and/or findings expressed are those of the author and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government. **Author contributions:** L.C. conceived the initial idea and designed the architecture, the standard, and the ontology. M.C. developed the XDL/ChemIDE and SynthReader software. S.H.M.M. helped develop the software to run the system and demonstrated the SynthReader with lidocaine, DMP, and two additional syntheses. A.I.L. demonstrated the SynthReader with AlkylFluor and six additional syntheses. G.K. helped develop and maintain the Chempiler software. The manuscript was written by L.C. together with M.C. and S.H.M.M. with input from all the authors. **Competing interests:** L.C. is the founder of DeepMatter Group PLC and Chemify Ltd., which aims to commercialize various aspects of the digitization of chemistry and synthesis using robotic platforms. L.C., M.C., S.H.M.M., and A.I.L. are listed as inventors on the UK patent 2006243.6, which describes this system. **Data and materials availability:** The ChemIDE app is available online (39). We have also made available the source code (40), as well as benchmarking results, literature survey data, and the XDL XML schema (41).

References and Notes

1. K. C. Nicolaou, Proc. R. Soc. A. 470, 20130690 (2014).
2. J. J. Irwin, B. K. Shoichet, J. Chem. Inf. Model. 45, 177–182 (2005).
3. O. J. Plante, E. R. Palmacci, P. H. Seeberger, Science 291, 1523–1527 (2001).
4. J. Li et al., Science 347, 1221–1226 (2015).
5. H. Saneii, M. R. Boroomand, M. R. Ferriell, U.S. Patent 5,746,982 (1998).
6. I. W. Davies, Nature 570, 175–181 (2019).
7. Reaxys (Elsevier, 2019); www.reaxys.com
8. S. Barata-Vallejo, S. Bonesi, A. Postigo, Org. Biomol. Chem. 14, 7150–7182 (2016).
9. M. M. Morgan, W. E. Piers, Dalton Trans. 45, 5920–5924 (2016).
10. D. Xu et al., J. Am. Chem. Soc. 142, 5785–5792 (2020).
11. B. Mahjour, Y. Shen, W. Liu, T. Cernak, Nature 580, 71–75 (2020).
12. V. Tshitoyan et al., Nature 571, 95–98 (2019).
13. P. Murray-Rust, Nature 451, 648–651 (2008).
14. R. G. Bergman, R. L. Danheiser, Angew. Chem. Int. Ed. 55, 12548–12549 (2016).

15. C. W. Coley, R. Barzilay, T. S. Jaakkola, W. H. Green, K. F. Jensen, ACS Cent. Sci. 3, 434–443 (2017).
16. C. Mellot-Draznieks, J. Dutour, G. Férey, Angew. Chem. Int. Ed. 43, 6290–6296 (2004).
17. J. Ghasemi, S. Saaidpour, Chem. Pharm. Bull. 55, 669–674 (2007).
18. S. Steiner et al., Science 363, eaav2211 (2019).
19. T. Bray et al., "Extensible Markup Language (XML)" (W3C, 2006).
20. A. C. Vaucher et al., ChemRxiv [Preprint] (2020); https://doi.org/10.26434/chemrxiv.11448177.v2. 21. E. Kim et al., J. Chem. Inf. Model. 60, 1194–1201 (2020).
22. L. Hawizy, D. M. Jessop, N. Adams, P. Murray-Rust, J. Cheminform. 3, 17 (2011).
23. T. Young, D. Hazarika, S. Poria, E. Cambria, IEEE Comput. Intell. Mag. 13, 55 75 (2018).
24. P. Corbett, P. Murray-Rust, in International Symposium on Computational Life Science, M. R. Berthold, R. C. Glen, I. Fischer, Eds. (Springer, 2006), pp. 107–118.
25. Materials and methods are available as supplementary materials.
26. L. A. Slaughter, A. D. Patel, J. L. Slaughter, J. Child Neurol. 28, 351–364 (2013).
27. T. J. Reilly, J. Chem. Educ. 76, 1557 (1999).
28. R. E. Ireland, L. Liu, J. Org. Chem. 58, 2899 (1993).
29. R. K. Boeckman Jr., P. Shao, J. J. Mullins, Org. Synth. 77, 141 (2000).
30. L. Kürti, B. Czakó, Strategic Applications of Named Reactions in Organic Synthesis: Background and Detailed Mechanisms (Elsevier Academic, 2005).
31. P. J. Stevenson, A. B. Treacy, M. Nieuwenhuyzen, J. Chem. Soc. Perkin Trans. 2 1997, 589–592 (1997).
32. G. Tojo, M. Fernández, Oxidation of Alcohols to Aldehydes and Ketones: A Guide to Current Common Practice (Springer, 2010).
33. M. Frigerio, M. Santagostino, S. Sputore, J. Org. Chem. 64, 4537–4538 (1999).
34. N. A. Reed, R. D. Rapp, C. S. Hamann, P. G. Artz, J. Chem. Educ. 82, 1053 (2005).
35. The original literature procedure (29) uses a similar validation method, except with excess benzyl alcohol. Because detailed steps were not provided, we used a similar literature procedure (34) based on the oxidation of menthol instead. Our method of validation was not aimed at approximating how DMP is used in practice.
36. N. W. Goldberg, X. Shen, J. Li, T. Ritter, Org. Lett. 18, 6102–6104 (2016).
37. P. Tang, W. Wang, T. Ritter, J. Am. Chem. Soc. 133, 11482–11484 (2011).
38. Royal Society of Chemistry, ChemSpider SyntheticPages (2019); http://cssp.chemspider.com/.
39. ChemIDE; https://croningroup.gitlab.io/chemputer/xdlapp/ 40. S. H. M. Mehr, M. Craven, A. I. Leonov, G. Keenan, L. Cronin, Code for XDL, Synthreader and ChemputerXDL, Zenodo (2020); doi:10.5281/zenodo.3955103.
41. S. H. M. Mehr, M. Craven, A. I. Leonov, G. Keenan, L. Cronin, Benchmarking results and the XDL XML schema, Zenodo (2020); doi:10.5281/zenodo.3955107.