Expert-level protocol translation for self-driving labs

Yu-Zhe Shi*, Fanxu Meng*, Haofei Hou*, Zhangqian Bi, Qiao Xu,
Lecheng Ruan✉, Qining Wang✉

Department of Advanced Manufacturing and Robotics,
College of Engineering, Peking University
*Equal contribution ✉ ruanlecheng@ucla.edu, qiningwang@pku.edu.cn

Abstract

Recent development in Artificial Intelligence (AI) models has propelled their application in scientific discovery, but the validation and exploration of these discoveries require subsequent empirical experimentation. The concept of self-driving laboratories promises to automate and thus boost the experimental process following AI-driven discoveries. However, the transition of experimental protocols, originally crafted for human comprehension, into formats interpretable by machines presents significant challenges, which, within the context of specific expert domain, encompass the necessity for structured as opposed to natural language, the imperative for explicit rather than tacit knowledge, and the preservation of causality and consistency throughout protocol steps. Presently, the task of protocol translation predominantly requires the manual and labor-intensive involvement of domain experts and information technology specialists, rendering the process time-intensive. To address these issues, we propose a framework that automates the protocol translation process through a three-stage workflow, which incrementally constructs Protocol Dependence Graphs (PDGs) that approach structured on the syntax level, completed on the semantics level, and linked on the execution level. Quantitative and qualitative evaluations have demonstrated its performance at par with that of human experts, underscoring its potential to significantly expedite and democratize the process of scientific discovery by elevating the automation capabilities within self-driving laboratories.

1 Introduction

The evolution of AI techniques has significantly accelerated the processes inherent to scientific discovery, with a notable impact observed within the domain of experimental sciences (Wang et al., 2023b). This influence is manifested through a variety of avenues: the generation of hypothesis spaces informed by extensive literature analysis (Jablonka et al., 2022; Kim et al., 2024), the interpretation of observational data via the identification of high-dimensional correlations (Jumper et al., 2021; Abramson et al., 2024), the engineering of novel structures that meet predefined specifications (Grisoni et al., 2021; Park et al., 2023), and the implementation of comprehensive simulations to ascertain the characteristics of potential products (Hie et al., 2021; Singh et al., 2023).

However, the findings facilitated by AI-driven research require further validation and exploration via empirical experiments, and may even entail a cyclical process where AI-generated hypotheses are refined based on the outcomes of real-world experiments, which demands the assembly of a sizable cohort of experienced experimenters to carry out these investigations in accordance with established *protocols* (McNutt, 2014). Unfortunately, the formation and sustenance of such a dedicated experimental cadre are fraught with considerable financial demands, and the collaborative engagement between people oriented towards AI methodologies and those grounded in experimental sciences is frequently encumbered by the communication gaps between distinct intellectual paradigms (Baker, 2016; Freedman et al., 2015; Munafò et al., 2017; Baker, 2021; Shi et al., 2023a).

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

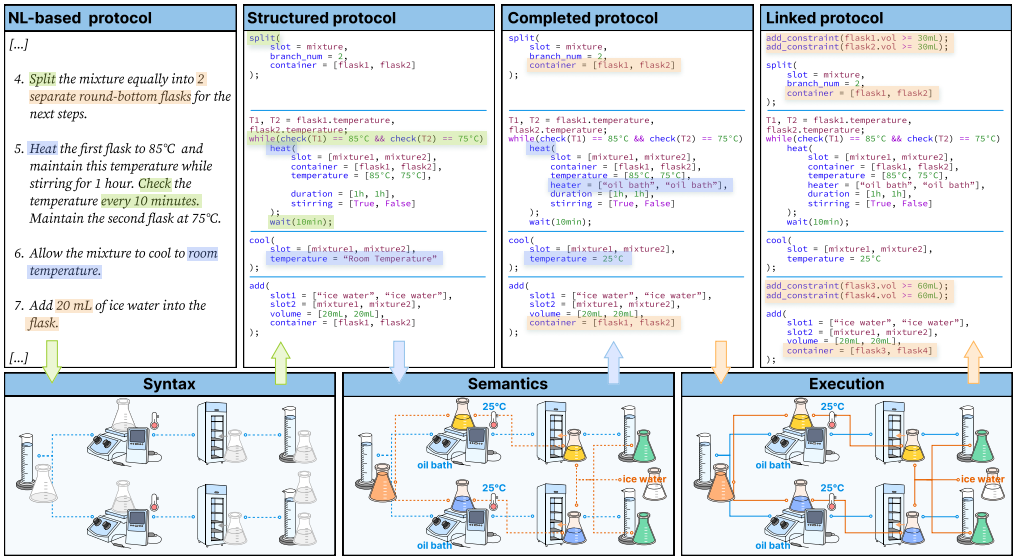

Figure 1: **Illustration of the protocol translation problem.** An NL-based protocol is translated to a **structured protocol**, then to a **completed protocol**, and finally to a **linked protocol** that is ready for self-driving laboratories along with a corresponding PDG, after being processed through the **syntax**, **semantics**, and **execution** levels. The three colors of arrows and text/ code highlights indicate the three translation steps respectively.

To bridge the aforementioned gap, the paradigm of self-driving laboratories has garnered attention, which automates experimental protocols via robotic systems, potentially revolutionizing the way experiments are conducted (Bédard et al., 2018; Steiner et al., 2019; Mehr et al., 2020; Rohrbach et al., 2022; Burger et al., 2020; Szymanski et al., 2023). Despite the promising outlook, designing such labs relies largely on the translation of protocols, primarily designed for human experimenters, into machine-readable instructions. This translation process necessitates extensive collaboration between domain experts, who possess the requisite scientific knowledge; and information technology specialists, who encode this knowledge into software and hardware systems. The inherently labor-intensive nature of such translation significantly prolongs the development of self-driving laboratories. The primary challenges are rooted in the discrepancies across three critical aspects (see Fig. 1):

Syntax Human experimenters can effortlessly **comprehend** protocols articulated in Natural Language (NL), whereas automated systems frequently necessitate dedicated syntax parsers to convert these protocols into a sequence of actionable steps. Consider the protocol: *"Split the mixture equally into 2 separate 50 mL round-bottom flasks for the next steps."* This example highlights the meticulous control over experimental procedures, explicitly directing the *"split"* of the mixture into precisely measured volumes — a crucial factor for achieving uniform outcomes in subsequent reactions. It is imperative at this level to uphold a **structured** representation of the mapping of operation conditions and the control flows of operations.

Semantics Human experimenters can **infer** implicit knowledge and context relying on the flexibility and adaptability of human understanding. In contrast, machine instructions necessitate a level of precision and rigidity that human communication does not inherently require. For instance, consider the protocol: *"Stir the mixture at room temperature for 5 minutes."* While a human expert might inherently understand that *"room temperature"* denotes a temperature range of 20-25 $^{\circ}C$ drawing on their prior knowledge, an automation system necessitates explicit information regarding such implicit details, which therefore need to be **completed** before execution.

Execution Human experimenters can **simulate** possible intermediate states and outcomes by considering the cumulative effects of a sequence of actions. For instance, given the two instructions adjacently: *"Add 35 mL water to the flask"* and *"Add 25 mL water to the flask"*, an experimenter can deduce that the flask's minimal capacity comes over 60 mL to prevent errors. For an automated system to perform a similar function, the actions need to be **linked** along their execution order.

Great efforts have been made on such translation tasks, among which Chemputer is representative (Mehr et al., 2020). This algorithm parses the NL-based protocol into XDL, a Domain-Specific Language (DSL) specially designed to describe chemical synthesis reactions. The completeness and

linkages are constructed with a set of manually-written constraints, with which the correctness of protocols can be further checked. This methodology has gained widespread acceptance in automated chemical synthesis, as a testament to the intensive efforts by domain and IT experts in developing XDL and the corresponding constraints. However, the application of a similar framework in other domains of experimental sciences, such as *Genetics*, *Medicine*, *Ecology*, and *Bioengineering*, would necessitate repeating these labor-intensive tasks on a case-by-case basis, thus underscoring the critical need for a more generally applicable, human-free protocol translator.

In this work, we propose a novel framework of human-free translator, designed to potentially facilitate applications across diverse experimental science domains without requiring extensive manual intervention. This framework decomposes the translation challenge into three hierarchical stages: structured on the syntax level, completed on the semantics level, and linked on the execution level, mirroring the cognitive steps undertaken by human experts in similar translation tasks. In the proposed work, the DSL, its constraints, and linkages are generated automatically, based on protocols tailored for human experimenters, thereby eliminating the need for labor-intensive manual processes.

Our contributions are threefold: (i) We conduct a systematic analysis of the existing discrepancies in protocol translation between human experimenters and automated systems in self-driving laboratories. From this analysis, we derive design principles that emulate human cognitive processes involved in protocol translation (Sec. 2). (ii) We devise an autonomous protocol translator through a tripartite framework that incrementally constructs PDGs, encapsulating the spatial-temporal dynamics of protocol execution across syntax, semantics, and execution levels (Sec. 3). (iii) Through both quantitative and qualitative evaluations in various experimental science domains, we demonstrate that our translator, when integrated as an auxiliary module for Large Language Models (LLMs), approaches the efficacy of skilled human experimenters and substantially surpasses the performance of purely LLMs-based alternatives in protocol translation tasks (Sec. 4).

2 Protocol translation for self-driving laboratories

In this section, we explore the translation of protocols for human experimenters to those suitable for self-driving laboratories. We analyze the task requirements across syntax (Sec. 2.1), semantics (Sec. 2.2), and execution (Sec. 2.3) levels. We pinpoint challenges at each level for both humans and machines, delving into systematic methods for addressing these issues. Leveraging expert insights, we delineate fundamental design principles for achieving effective protocol translation (Sec. 2.4).

2.1 Syntax level

Operation-condition mapping In NL-based protocols, operations and their corresponding parameters such as input reagents and conditions, are entangled with each other. For example, *"Dissolve 10 g of sodium chloride in 100 mL of distilled water at 80°C"*, the entanglements of actions and conditions highlight the complexity machines face in parsing such protocols. Human experimenters can *recognize* them without information loss thanks to the *internalized language* for parsing NL (Chomsky, 1956, 2007). In contrast, protocols for machines must be represented precisely, with proper extraction of keys and values, and matching between them with appropriate data structures.

Operation control flows In NL-based protocols, both linear and non-linear control flows are implicitly embedded in the text. While linear control flows, *i.e.*, workflows in sequential execution order, can be straightforward, non-linear control flows such as iterative loops and parallel operations can be hard to detect because the signal and the operational domain can be separated. Consider the protocol: *"Repeat the titration until the endpoint is reached, then record the volume of titrant used"*. These steps embody a non-linear control flow, challenging machines to correctly interpret the iterative process involved. Even human experts have to read the protocols carefully to understand the local and global structures to match the signals with operational domains, let alone machines.

2.2 Semantics level

Latent semantics of known unknowns Some assigned values of parameters are regarded as common sense knowledge of domain experts by default, thus the values are omitted for simplicity or referred to via a *proxy name* following the domains' conventions. For example, the protocol instruction *"Dry the purified product at room temperature"* relies on the experimenter's understanding

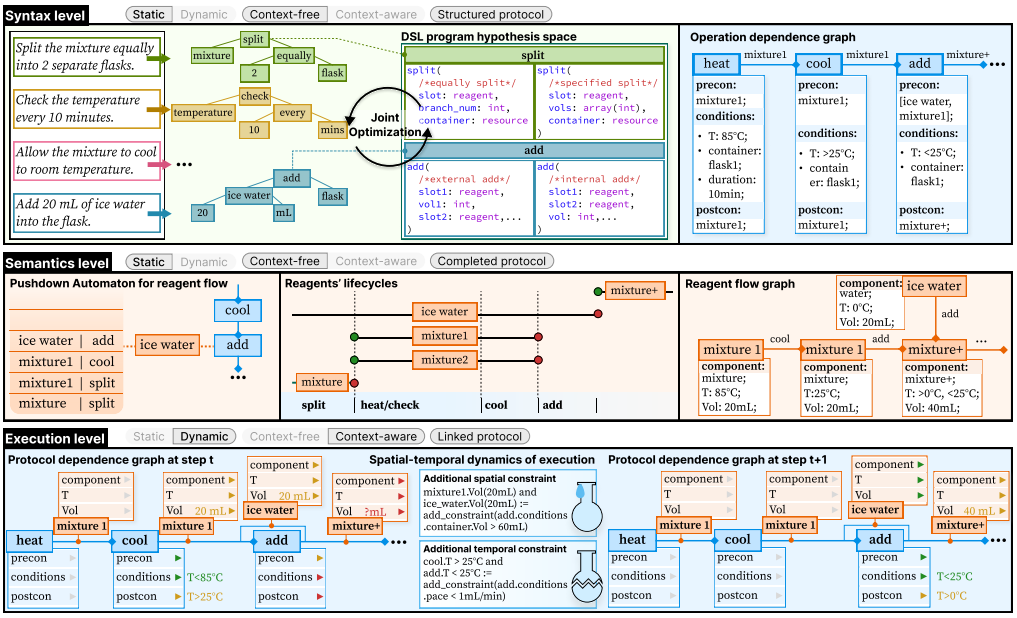

Figure 2: **The design principles and the resulting pipeline of our translator. (Syntax level)** Operation dependence synthesis on the syntax level, through the joint optimization of DSL program syntax space and the parsing tree of the NL-based protocols. This process is static and context-free. **(Semantics level)** Reagent flow analysis on the semantics level, through an automaton scheme maintaining the lifecycles of reagents and intermediate products. This process is static and context-free. **(Execution level)** Spatial-temporal dynamics analysis on the execution level, through the partial execution trace model based on the spatial-temporal dual constraint representation. This process is dynamic and context-aware.

of what constitutes *"room temperature"*. However, machines substantially suffer from such latent semantics, implying that every value of parameters should be made explicit.

Latent semantics of unknown unknowns Sometimes, even required parameters for a specific operation are omitted from the protocols either or not intentionally, causing *unknown unknowns* that one may even be not aware of the absence of such information. For instance, the protocol instruction *"Centrifuge the sample after adding the enzyme"* does not specify the key controlling parameter, speed or duration, for the *"centrifuge"* operation, before describing its specific value. Both human and machines require every parameter of operations to be grounded.

2.3 Execution level

Capacity of resources Protocols often omit explicit specifications of resource capacities, leading to potential execution errors like exceeding a device's maximum capacity. This issue, inherent in the execution sequence, is undetectable by analyzing single operations alone. For example, the instruction *"Transfer the mixture to a beaker"* requires choosing a beaker with adequate capacity, a decision based on the cumulative volume from previous steps. Humans intuitively manage this through a mental simulation of the experimental process (Gallese and Goldman, 1998). Machines, therefore, need a pre-execution verification mechanism to ensure resource capacities are not exceeded, highlighting the need for an integrated understanding of the experimental sequence.

Safety of operations In addition to managing resource capacities, another source of runtime errors stems from operations that, while semantically valid, may lead to adverse or dangerous outcomes in certain execution contexts. Such scenarios necessitate a *dual-constraint* approach, where experimenters are mindful not only of the actions required *what I should do* but also of potential missteps to avoid *what I must not do*. For instance, the instruction *"Heat the reaction mixture to 70°C"* can be appropriate or hazardous, depending on the mixture's composition — safe with a heat-stable catalyst, but risky with a heat-sensitive component due to potential decomposition. To navigate these complexities, human experts effectively run mental simulations, conducting *"What if?"* queries and counterfactual reasoning (Hoch, 1985) to anticipate the consequences of their actions. Similarly, machines need a system to draw upon domain-specific knowledge and historical context to assess the safety of each operation, ensuring that all actions are contextually appropriate and safe.

2.4 Design principles inspired by human experimenters

Human experts' cognitive capabilities on the translation of protocols serve two key roles: understanding protocols for in-hand experiments and manually developing translators for self-driving laboratories. Inspired by these practices, we outlined design principles for our translator and assessed the strengths and weaknesses of current DSLs for NL-based protocols, such as XDL (Steiner et al., 2019), ULSA (Wang et al., 2022), ORD (Kearnes et al., 2021), Biocoder (Ananthanarayanan and Thies, 2010), Autoprotocol (Strateos, 2023), and the family of DSLs (hereinafter called ADSL) which are automatically designed by the AutoDSL tool driven by domain corpora (Shi et al., 2024a).

Operation dependence synthesis for the syntax level To precisely comprehend the complicated operation-condition mappings and non-linear control flows, machines should equip with an *externalized language* in parallel with humans' *internalized language* (Chomsky, 2007). A machine-recognizable language commonly possesses a Context-Free Grammar (CFG) which externally defines the key-value structures on different hierarchies: (i) operation as key, reagents and conditions as values; (ii) condition as key, the corresponding parameters as values; and (iii) signal of control flow as key, the corresponding operational domains as values. If a protocol can be parsed into an Abstract Syntax Tree (AST) with the CFG, it is verified on the syntax level (Hopcroft et al., 1996), resulting in the dependency structures of the operation flow (see Fig. 2 Top). All DSLs mentioned before are context-free languages with CFGs (Fowler, 2010), echoing this design principle.

Reagent flow analysis for the semantics level Despite the merits of DSLs based on CFGs, the context-free nature hinders verification on the semantics level, which is pivotal in protocols essentially describing procedures, where the preconditions and postconditions between temporally adjacent operations can be end-to-end connected. To be specific, although a CFG defines a structural space with hierarchies of operations, conditions, parameters, and control flows, *i.e.*, *"There must be several parameters corresponding to a condition"*, it does not constrain the mappings under the context of domain-specific knowledge, *i.e.*, *"There are parameters controlling the Temperature, Duration, and Acidity of the condition"*. If the exact mapping between keys and values cannot be specified, the self-driving laboratories can hardly be aware of the loss of completeness, *i.e.*, being aware of the omitted conditions given an operation or the missing value given a parameter, due to the extremely large search space over all symbols given by the corresponding DSLs (Gulwani et al., 2017). The design choices of DSLs diverge on this level, where Autoprotocol only supports verification on the syntax level and does not possess any domain knowledge, ORD and ULSA offer the relations between operations and conditions without more fine-grained parameters, while XDL, Biocoder, and ADSL offer the find-grained key-value relation below the hierarchy of operations without more constraints about the values, *e.g.*, suggested values of specific parameters. Hence, we require a mechanism for completing the structures of reagent flow (see Fig. 2 Middle), while the completeness of the fine-grained parameters is guaranteed by the DSLs.

Spatial-temporal dynamics analysis for the execution level Completion on the semantics level is conducted statically because the semantics of operations are viewed individually rather than contextualized in the execution sequence. Regrettably, such effort cannot guarantee that the protocols can be executed successfully without any errors in the run time, which is unacceptable by self-driving laboratories (Christensen et al., 2021; Seifrid et al., 2022). One way is to have domain experts write down all of the potential bad cases as constraints and use them for verification. However, run-time errors raised in the dynamic context of operations are heavily long-tail distributed. This makes it extremely hard to predict such errors from statistical *hindsight* (Pearl, 2019), *i.e.*, the set of collected post-hoc bad cases. Thus, we leverage the powerful *foresight* based on simulation, which spans the full probabilistic worlds of each operation by its semantic constraints, both on the *spatial* dimension, *e.g.*, capacity of resources captured by the reagent dependency, and the *temporal* dimension, *e.g.*, safety of operations captured by the operation dependency. The simulation is conducted along the topological order of the corresponding execution flow graph. At each operation unit, both historical operations in the same protocols and similar operations in other protocols are recalled dynamically, checking and refining the dual-constraint spaces accordingly (see Fig. 2 Bottom). Interestingly, none of the DSLs in our discussion take this feature as part of language design and only XDL employs an external compiler with hand-crafted rules for error detection. Such consideration is reasonable because in mainstream DSL design, verification on the execution level is not guaranteed by the DSLs themselves for design simplicity and user convenience (Mernik et al., 2005). Consequently, we require an environment to dynamically check the correctness of execution both spatially and temporally, through synthesizing operation and reagent dependencies.

3 The framework of protocol translation

In this section, we introduce the three-stage framework for human-free protocol translation, which gradually constructs a structural representation of protocols, called the *Protocol Dependence Graph* (PDG). The PDG makes explicit both the operation and reagent dependencies for a protocol. Operation dependence echoes the concept of program control flow, which derives the condition of sequential, branch, or loop execution of protocol operations (Sec. 3.1). Reagent flow provides an explicit representation of the reagent instantiate-exploit relationships implicitly in the protocol (Sec. 3.2). Additionally, we simulate the protocol execution process using the PDG, checking and refining operation sequences under the spatial and temporal dynamics of execution (Sec. 3.3).

3.1 Operation dependence synthesis for the syntax level

The operation dependence models the topological order for executing operations in a protocol. The procedure is executed sequentially from the first operation in the protocol to the last, unless the experimenter encounters structures that change the execution flow, such as branches and loops. In practice, we extract the operation dependence by compiling the protocol to DSL programs.

Input The compilation process is conducted based on the corresponding DSL $\mathcal{L} = \{\mathcal{S}, \Lambda\}$. The CFG-based syntax $\mathcal{S} = (S, V, \Sigma, R)$ includes (i) the start symbol S; (ii) the variable set $V = \{V_{\texttt{ctrl}}, V_{\texttt{op}}, V_{\texttt{cond}}, V_{\texttt{par}}\}$ with placeholders for control flow signals $V_{\texttt{ctrl}}$, operations $V_{\texttt{op}}$, conditions $V_{\texttt{cond}}$, and parameters $V_{\texttt{par}}$; (iii) the set of terminals Σ, which are the grounded values of parameters; (iv) the set of production rules R defining the structural space between the four variable sets. The semantics $\Lambda = (T_{\texttt{ctrl}}, T_{\texttt{op}}, T_{\texttt{cond}}, T_{\texttt{par}}, T_R)$ constrains the variables and production rules, assigning the placeholders with substantial meanings. To note, according to the design choice discussed in Sec. 2.4, the DSL used here should be one of XDL, Biocoder, and ADSL, *i.e*. the DSLs with the most fine-grained structural representation compared with their counterparts.

Pre-processing Given an input protocol $\mathbf{c}$ for translation, we first parse the NL sentences by an off-the-shelf tool and extract the actions accordingly. Then, the extracted actions are matched with the operations $o \in T_{\texttt{op}}$ of the DSL, according to both exact match score and semantic similarity. Afterwards, we extract the arrays of entities related to the extracted action $\mathbf{e}_t \in \mathcal{E}$ by an off-the-shelf tool, where we regard the output labels to the entities and relations as *pseudo-labels* because they can possibly be noisy. Please refer to Appx. C.1 for implementation details.

DSL program synthesis Synthesizing structural representation given unstructured signal is challenging (Billard, 2000, 2006). Specifically, the one-to-many mappings of many DSL operations bring uncertainty into the matching. For example, the operation `add` possesses distinct patterns: two or three input slots. This further distorts the matching of reagents, conditions, and parameters because off-the-shelf tools can hardly detect the exact categories of these entities deeply rooted in domain-specific knowledge. Since the observation and hypothesis spaces are both noisy, we propose to jointly optimize the patterns of operations and the pseudo-labels. We denote the set of all possible program patterns generated by operation o as $o^* = \{\texttt{p} | o \Rightarrow^* \texttt{p}, T_R \Rightarrow^* \texttt{p}, \texttt{p} \in T^*_{\texttt{ctrl}} \cup T^*_{\texttt{op}} \cup T^*_{\texttt{cond}} \cup T^*_{\texttt{par}}\}$. A synthesized DSL program is defined as $\texttt{p}(\mathbf{c}) = \langle \texttt{p}(o_1), \texttt{p}(o_2), \ldots, \texttt{p}(o_{|p(\mathbf{c})|}) \rangle$, where $\texttt{p}(o_t)$ is the program of an operation assembled with its corresponding conditions and parameters under the selected pattern in o^*. Let $s(\mathbf{c}) = \langle \mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{|s(\mathbf{c})|} \rangle$ represent the sequence of operation-related entities, the objective of optimization can be

$$\arg \min_{\texttt{p}(o_t), \mathbf{e}_t} \sum_t \sum_{o_t^*} D\big(\texttt{p}(\mathbf{c}) \| s(\mathbf{c})\big), \tag{1}$$

where $D(\cdot\|\cdot)$ is a divergence function with three indicators: (i) the selected pattern examples should be as close as possible to the text span; (ii) the selected pattern should be as similar as possible with the extracted subject-verb-object structure; and (iii) as many labeled entities as possible should be mapped to the parameter space (see Fig. 3B). Though $|o^*|$ is not a large value, the whole sequence of operations can yield an exponential complexity. Hence, to make the joint optimization tractable, we separate the search of solution into two steps in the spirit of Expectation Maximization (EM): (i) Expectation: sampling programs from the legal space defined by the corresponding DSL, with a program-size-sensitive prior $|\texttt{p}(o_t)|^{-1}$; (ii) Maximization: randomly alternating symbols in $s(\mathbf{c})$ by matching them with those in $\texttt{p}(\mathbf{c})$ and greedily select the edits that decrease the objective function.

3.2 Reagent flow analysis for the semantic level

The DSL programs are further contextualized by associating operations with reagent flow. Reagent flow indicates the transfer of reagents among operations, reflecting how one operation impacts the subsequent ones. We define the reagent flow following the *reaching definitions* schema (Alfred et al., 2007), which is commonly used to capture the life cycle of a variable in compiler design. This schema determines a set of reagents reachable at each point in a protocol, and subsequently tracks the *kills* and *defines* of an operation, *i.e.*, whether a reachable reagent is consumed, or a new reagent is yielded, in an operation.

Algorithm 1 Reagent flow analysis

procedure TRANSITION(M, o)
  ▷ *Context Transition*
  ERASE($M(\Gamma)$, KILLS($M(\Gamma), o$))
  APPEND($M(\Gamma)$, DEFINES(o))
  ▷ *State Transition*
  $M(q) \leftarrow (M(q) \setminus o) \cup$ NEXTOPS(o)
procedure FLOW($p(\mathbf{c})$, M)
  $R = \{\}$ ▷ *Set of reagent dependence*
  $M(q) \leftarrow \{o_1\}$ ▷ *Initial State*
  $M(\Gamma) \leftarrow \langle\rangle$ ▷ *Initial Memory*
  while $M(q) \neq \phi$ **do**
    TRANSITION(M, $M(q)$)

Input We denote the reagents consumed and the intermediate products yielded by each operation o as IN(o) and OUT(o) respectively. The objective is to find a set of operation pairs $\{\langle o_i, o_j\rangle \,|\, \text{OUT}(o_i) \cap \text{IN}(o_j) \neq \phi\}$ such that OUT(o_i) is required as input by IN(o_i).

The reaching definitions schema We determine the availability of a reagent at each step by locating where it is defined in a protocol when execution reaches each operation. A reagent r reaches an operation o, if there is a path from the point following r to o, such that r is not *killed*, *i.e.*, consumed, along that path. Any reagent r that reaches an operation o might be killed at that point, and o may yield new intermediate products r' that reach future operations. Notably, according to statistics on corpora of protocols (Vaucher et al., 2020), for about 90% operations of a target DSL, if $o_i \prec o_j$ are two adjacent operations, $\text{OUT}(o_i) \cap \text{IN}(o_j) \neq \phi$ holds. This implies that the reagent generated by preceding operation is likely to be used and then be killed instantly by the following operation.

Reagent flow analysis via operation flow traversal We traverse the DSL program in execution order to leverage the reagent locality revealed from statistical results, determining the reachability and life cycle of reagents. A Pushdown Automaton (PDA) with a random access memory is adopted to record reachable reagents as operation context, defining and killing reagents at each operation point along the computation[1]. A PDA is formally defined as a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, where $Q \subseteq T_{\text{op}}$ indicates the set of states, $\Sigma = T_{\text{op}}$ represents the domain of inputs, $\Gamma \subseteq T_{\texttt{par}}$ denotes possible memory elements, $\delta \subseteq Q \times \Sigma \times \Gamma \rightarrow Q \times \Gamma^*$ is the transition procedure (TRANSITION in Alg. 1), $q_0 = o_1$ defines the initial state, $Z = \epsilon$ is the initial memory element, and F denotes the set of accepting states. The reagent dependence construction process (FLOW in Alg. 1) traverses the DSL program in execution order by leveraging the NEXTOPS utility, which evaluates to subsequent operations. In every transition step with input, the killed reagents are removed from the memory, and the defined reagents are added to the memory. After a reagent is killed, the pair of the operations that defined it and killed it will be added to the set of reagent flow constraints. The accepting state is reached if the memory is empty at the end of execution, *i.e.*, all reagents defined in operations are killed by other operations. We employ state-of-the-art LLMs to extract reagent entities from NL-based protocol descriptions for the two utilities KILLS and DEFINES through instruction-following in-context learning (Wei et al., 2021; Brown et al., 2020) (refer to Appx. C.2 for details).

3.3 Spatial-temporal dynamics for the execution level

While the pre-specified PDG analysis indicates the things *should* be done to follow operation and reagent flow, we still need to care about the things *must not* be done by describing the activities that may be performed and the constraints prohibiting undesired execution behavior. Therefore, we introduce a constrained-based execution model to support dynamic protocol simulation, getting grounded in the theories of process modeling and execution (Dourish et al., 1996; Pesic et al., 2007).

A constraint-based protocol execution model We extend the DSL program $\texttt{p}(\mathbf{c})$ by a constraint set $C = C_{op} \cup C_{reg} \cup C_s \cup C_t$ to construct a constraint-based execution model $S = (\texttt{p}(\mathbf{c}), C)$. The execution of a program is represented by a trace $\sigma = \langle (o_1, c_1), (o_2, c_2), \ldots, (o_{|\texttt{p}(\mathbf{c})|}, c_{|\texttt{p}(\mathbf{c})|})\rangle$,

[1]It is worth noting that this extended PDA with random access can be shown to be in the same computation class as Turing machines (Aho and Ullman, 1972), and we employ this extended PDA due to its simplicity.

where the order of o_i reflects the temporal sequence. The execution context c_i defines the spatial environment in which each operation is performed. Each constraint $c \in S(C)$ is a predicate that maps the execution trace σ to a binary condition denoting satisfy or not. An execution trace σ is said to satisfy the program constraint if and only if $|\sigma| = |\mathtt{p}(\mathbf{c})|$ and $c(\sigma)$ holds for all $c \in C$.

Leveraging partial execution trace for spatial and temporal constraints Explicit constraints, namely operation and reagent flow, are easy to satisfy. Unfortunately, deriving implicit constraints, *e.g.*, the capacity of resources and safety of operations, is case-by-case for each protocol, requiring expert efforts. We propose profiling the context through execution to derive implicit spatial and temporal constraints that meet domain-specific requirements. An execution trace is defined as *partially* satisfying the constraints C if it follows the operation and reagent flow; that is, for any $\langle o_i, o_{j>i} \rangle$ in trace σ, there exists at least a pair of valid operation flow path and reagent flow path from o_i to o_j.

4 Results

In this section, we compete our framework with human experts on the overall translation task, and assess the utility of each component of the framework by comparing with alternative approaches.

4.1 Experimental setting

Materials We select 75 complicated experiments with 1,166 steps in total as the testing set, from the domains of *Chemical Synthesis* (235 steps in 10 experiments; 235 in 10 for simplicity; "Synthesis" for abbreviation), *Genetics* (396 in 34), *Medical and Clinical Research* (307 in 23, "Medical"), *Bioengineering* (218 in 17), and *Ecology* (10 in 1). Please refer to Appx. D for details.

Expert-created protocol translation We recruited five groups of experienced experimenters, each specializing in a different domain, with seven participants in each group. Every participating experimenter holds at least a Master's degree related to the corresponding domain, has obtained at least six years' experience in manually conducting pre-designed experiments of the domain, has acquired elementary programming skills, and has at least heard of self-driving laboratories. These human experts are asked to translate the original NL-based protocols for their domains into those suitable for self-driving laboratories. Their outputs are subjected to DSL-based representations and complete PDGs to evaluate machines' behaviors, which are clearly demonstrated by a running example and the examples in the DSL documentation. Outputs from experts are carefully cross-validated and the individual divergence between them is minimized through an expert-panel-driven workshop discussion following the established workflow (Reilly et al., 2023). Translation results of human experts and machines are serialized and are compared through ROUGE and BLEU metrics (Lin, 2004; Papineni et al., 2002). Please refer to Appx. B for ethics considerations.

Alternative methods We compare our translator with alternative methods on the first two levels, to investigate the effects of early stages on the overall translation result. On the syntax level, we compare the syntactic synthesis method (referred to as `Ours-SY`) with `ConDec-SY` (Wang et al., 2023a), which synthesizes DSL programs by LLMs with external DSL grammars as constraint, and a baseline `DSL-LLM-SY` leveraging the minimal realization used in Shi et al. (2024a). On the semantics level, we compare the deductive verification method (referred to as `Ours-SE`) with `NL-RAG-LLM-SE`, which retrieves on the embedded vector database of original NL-based protocols, and a baseline `NL-LLM-SE` implemented by pure prompt-engineering on LLMs. Since the I/Oes of all stages are unified in the pipeline, we implement an overall baseline `Best-Baseline` that combines the strongest alternative methods within the evaluation of the first two stages.

4.2 Overall assessment on expert-created protocol translation

Result Comparing the overall output of our translator and ideal human experimenters, we find that our translator approaches the level of experts with average performance higher than 85% across all indicators. Our translator significantly outperforms the alternative pipeline `Best-Baseline` on the ($t(148) = -17.71, \mu_d < 0, p < .0005$; see Fig. 3C).

Discussion We find that our translator demonstrates similar performance to human experts in translating protocols with complete parameters and clear descriptions (see Fig. 4A). However, in cases where the linear description of the experimental protocol is lacking, our translator and human

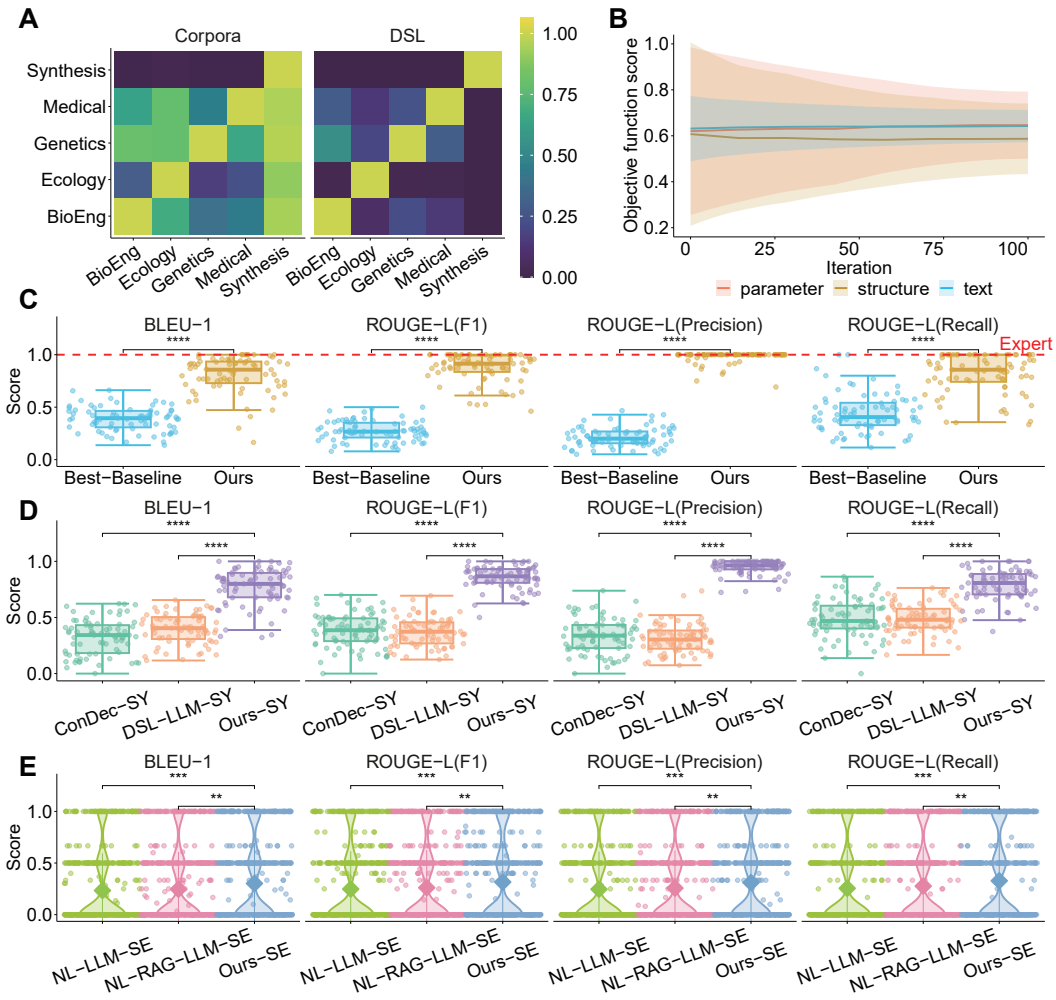

Figure 3: **Results of experiment.** **(A)** Distinctions between various domains regarding domain-specific corpora and the corresponding DSLs. **(B)** Convergence of the three indicators in the objective function for program synthesis. **(C)** Our translator significantly outperforms the best baseline and approaches human-level performance. **(D)** Our translator significantly outperforms alternative methods on the syntax level. **(E)** Our translator significantly outperforms alternative methods on the semantics level.

experts diverge. Specifically, our translator tends to translate based on the information within the sentence or between adjacent sentences, while human experts tend to consider the overall experimental process comprehensively. Though there are minor gaps, these observations suggest that our translator is approaching the level of performance of experienced human experimenters. Please refer to Appx. E.3 for case studies on the distinctions between the behaviors of experts and machines.

4.3 Comparison between alternative models

Result On the syntax level, our `Ours-SY` significantly outperforms alternative approaches ($t(148) = -17.07, \mu_d < 0, p < .0005$ for `ConDec-SY`; $t(148) = -15.47, \mu_d < 0, p < .0005$ for `DSL-LLM-SY`; see Fig. 3D). On the semantics level, our `Ours-SE` significantly outperforms alternative approaches ($t(148) = -2.52, \mu_d < 0, p < .05$ for `NL-RAG-LLM-SE`; $t(148) = -3.07, \mu_d < 0, p < .005$ for `NL-LLM-SE`; see Fig. 3E).

Discussion Compared with alternative methods on the syntax level, our translator excels in ensuring the accuracy of translations across different protocols thanks to the PDG representation, as shown in Fig. 4B. On the semantics level, our translator performs better in translating incomplete protocols with missing information than other baselines, as shown in Fig. 4C. We explain these merits with the properties of structural representation, which defines a representation space with

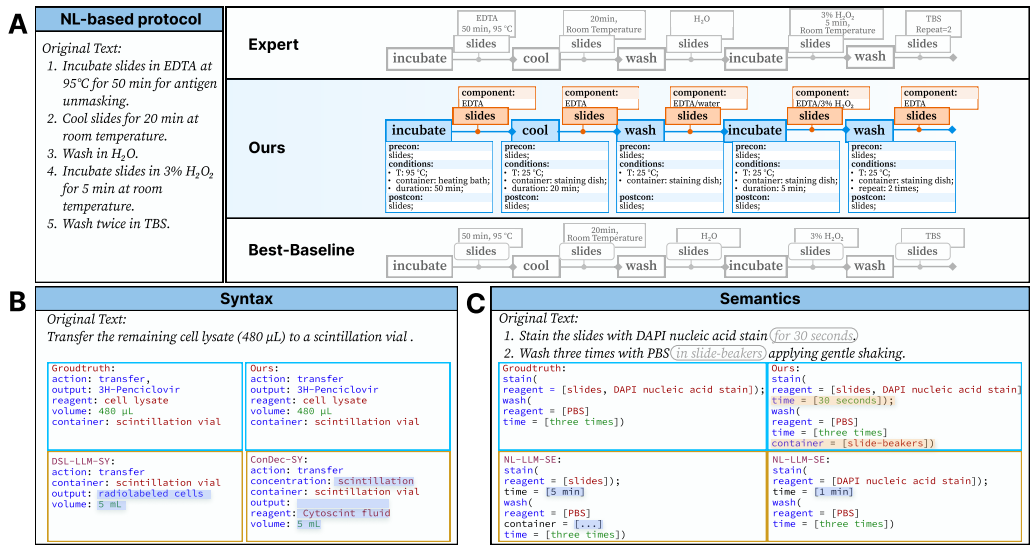

Figure 4: **Showcases of the results. (A)** Examples of the final PDGs generated by our translator, the alternative method, and human experts. **(B)** Examples of structured protocols output by our translator and alternative methods. **(C)** Examples of completed protocols output by our translator and alternative methods.

high expressive power (Felleisen, 1991; Lloyd, 2012), capturing information along the spectrum of granularity, from local details like fine-grained parameter-value relations to global-structure of control flows. By contrast, the original NL representation and its embedding space may not possess such extent of expressive power (Zhang et al., 2021). Consequently, mapping the protocols into a structural latent space for query or retrieval should be more suitable than directly operating on the original space. Please refer to Appxs. E.1 and E.2 for details on the utilities of different approaches.

5 General discussions

In this work, we study the problem of translating protocols for human experimenters into those proper for self-driving laboratories. We design a human-free protocol translator under the inspiration of human experts' cognitive processes in protocol translation. Results suggest that our translator is comparable with experienced human experimenters in protocol translation, further implying its potential to serve as a plug-and-play module for fully automated scientific discovery.

Closing the loop of automatic scientific discovery The realization of automatic protocol translation closes the *last-mile* of closed-loop scientific discovery. Integrating the translator with other cutting-edge techniques, we can expect such a pipeline in the future: after AI models for scientific discovery have output insights from initial observations, there is a self-driving laboratory ready for producing and testing the designs. The creation of this laboratory is also automatic (Shi et al., 2024c). In the conventional creation process, we must employ a cohort of experienced domain experts to hand-craft a DSL tailored for representing protocols of the target domain, and then hand-craft a translator with pre-defined production rules and constraints to translate the protocols into machine executables. Hence, the status quo has been changed: we can exploit the AutoDSL tool to obtain the DSL based on a corpus of the target domain (Shi et al., 2024a), and then set up our automatic translator for final execution; we can also integrate the DSLs and our translator into LLM-based "scientist agents" for a more reliable and flexible domain specification (Boiko et al., 2023; Bran et al., 2023). The integration of these approaches facilitates the grounding of AI for scientific discovery, which helps AI researchers test, produce and deploy their discoveries more seamlessly.

Valuing domain experts One might worry that the full realization of closed-loop scientific discovery would pose a severe impact on conventional experimental scientists. Indeed, AI for scientific discovery and self-driving laboratories do not compete with conventional scientific discovery. As interdisciplinary methodologies, they still demand the supervision of scientists for tacit knowledge, creativity, and integrity (Shi et al., 2024b). Also, such techniques free domain experts from those time-consuming and labor-intensive low-level workloads and focus them on high-level thinking.

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China under Grants 91948302 and 52475001. Part of the authors are visiting students at Peking University during this work. In particular, Z. Bi is visiting from Huazhong University of Science and Technology and Q. Xu is visiting from University of Science and Technology of China. The authors would also like to thank Jiawen Liu for her assistance in figure drawings.

References

Abramson, J., Adler, J., Dunger, J., Evans, R., Green, T., Pritzel, A., Ronneberger, O., Willmore, L., Ballard, A. J., Bambrick, J., et al. (2024). Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, pages 1–3.

Aho, A. V. and Ullman, J. D. (1972). *The theory of parsing, translation, and compiling*, volume 1. Prentice-Hall Englewood Cliffs, NJ.

Alfred, V. A., Monica, S. L., and Jeffrey, D. U. (2007). *Compilers Principles, Techniques & Tools*. pearson Education.

Ananthanarayanan, V. and Thies, W. (2010). Biocoder: A programming language for standardizing and automating biology protocols. *Journal of Biological Engineering*, 4:1–13.

Baker, M. (2016). 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604).

Baker, M. (2021). Five keys to writing a reproducible lab protocol. *Nature*, 597(7875):293–294.

Bédard, A.-C., Adamo, A., Aroh, K. C., Russell, M. G., Bedermann, A. A., Torosian, J., Yue, B., Jensen, K. F., and Jamison, T. F. (2018). Reconfigurable system for automated optimization of diverse chemical reactions. *Science*, 361(6408):1220–1225.

Billard, L. (2000). Regression analysis for interval-valued data. *Data Analysis, Classification, and Related Methods*.

Billard, L. (2006). Symbolic data analysis: what is it? In *Proceedings in Computational Statistics: 17th Symposium Held in Rome, Italy*.

Boiko, D. A., MacKnight, R., Kline, B., and Gomes, G. (2023). Autonomous chemical research with large language models. *Nature*, 624(7992):570–578.

Bran, A. M., Cox, S., Schilter, O., Baldassari, C., White, A., and Schwaller, P. (2023). Augmenting large language models with chemistry tools. In *NeurIPS 2023 AI for Science Workshop*.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems*.

Burger, B., Maffettone, P. M., Gusev, V. V., Aitchison, C. M., Bai, Y., Wang, X., Li, X., Alston, B. M., Li, B., Clowes, R., et al. (2020). A mobile robotic chemist. *Nature*, 583(7815):237–241.

Chomsky, N. (1956). Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124.

Chomsky, N. (2007). Approaching ug from below. *Interfaces + recursion = language*, 89:1–30.

Christensen, M., Yunker, L. P., Shiri, P., Zepel, T., Prieto, P. L., Grunert, S., Bork, F., and Hein, J. E. (2021). Automation isn't automatic. *Chemical Science*, 12(47):15473–15490.

Dourish, P., Holmes, J., MacLean, A., Marqvardsen, P., and Zbyslaw, A. (1996). Freeflow: mediating between representation and action in workflow systems. In *Proceedings of the 1996 ACM conference on Computer supported cooperative work*.

Felleisen, M. (1991). On the expressive power of programming languages. *Science of computer programming*, 17(1-3):35–75.

Fowler, M. (2010). *Domain-specific languages*. Pearson Education.

Freedman, L. P., Cockburn, I. M., and Simcoe, T. S. (2015). The economics of reproducibility in preclinical research. *PLoS Biology*, 13(6):e1002165.

Gallese, V. and Goldman, A. (1998). Mirror neurons and the simulation theory of mind-reading. *Trends in Cognitive Sciences*, 2(12):493–501.

Grisoni, F., Huisman, B. J., Button, A. L., Moret, M., Atz, K., Merk, D., and Schneider, G. (2021). Combining generative artificial intelligence and on-chip synthesis for de novo drug design. *Science Advances*, 7(24):eabg3338.

Gulwani, S., Polozov, O., Singh, R., et al. (2017). Program synthesis. *Foundations and Trends® in Programming Languages*, 4(1-2):1–119.

Hie, B., Zhong, E. D., Berger, B., and Bryson, B. (2021). Learning the language of viral evolution and escape. *Science*, 371(6526):284–288.

Hoch, S. J. (1985). Counterfactual reasoning and accuracy in predicting personal events. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 11(4):719.

Honnibal, M. and Johnson, M. (2015). An improved non-monotonic transition system for dependency parsing. In *Annual Conference on Empirical Methods in Natural Language Processing*.

Hopcroft, J. E., Motwani, R., and Ullman, J. D. (1996). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Longman Publishing Co., Inc.

Jablonka, K. M., Patiny, L., and Smit, B. (2022). Making the collective knowledge of chemistry open and machine actionable. *Nature Chemistry*, 14(4):365–376.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589.

Kearnes, S. M., Maser, M. R., Wleklinski, M., Kast, A., Doyle, A. G., Dreher, S. D., Hawkins, J. M., Jensen, K. F., and Coley, C. W. (2021). The open reaction database. *Journal of the American Chemical Society*, 143(45):18820–18826.

Kim, C., Gadgil, S. U., DeGrave, A. J., Omiye, J. A., Cai, Z. R., Daneshjou, R., and Lee, S.-I. (2024). Transparent medical image ai via an image–text foundation model grounded in medical literature. *Nature Medicine*, pages 1–12.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*.

Lloyd, J. W. (2012). *Foundations of logic programming*. Springer Science & Business Media.

McNutt, M. (2014). Reproducibility. *Science*, 343(6168):229–229.

Mehr, S. H. M., Craven, M., Leonov, A. I., Keenan, G., and Cronin, L. (2020). A universal system for digitization and automatic execution of the chemical synthesis literature. *Science*, 370(6512):101–108.

Mernik, M., Heering, J., and Sloane, A. M. (2005). When and how to develop domain-specific languages. *ACM Computing Surveys (CSUR)*, 37(4):316–344.

Munafò, M. R., Nosek, B. A., Bishop, D. V., Button, K. S., Chambers, C. D., Percie du Sert, N., Simonsohn, U., Wagenmakers, E.-J., Ware, J. J., and Ioannidis, J. (2017). A manifesto for reproducible science. *Nature Human Behaviour*, 1(1):1–9.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*.

Park, N. H., Manica, M., Born, J., Hedrick, J. L., Erdmann, T., Zubarev, D. Y., Adell-Mill, N., and Arrechea, P. L. (2023). Artificial intelligence driven design of catalysts and materials for ring opening polymerization using a domain-specific language. *Nature Communications*, 14(1):3686.

Pearl, J. (2019). The seven tools of causal inference, with reflections on machine learning. *Communications of the ACM*, 62(3):54–60.

Pesic, M., Schonenberg, M., Sidorova, N., and van der Aalst, W. M. (2007). Constraint-based workflow models: Change made easy. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*.

Reilly, J., Shain, C., Borghesani, V., Kuhnke, P., Vigliocco, G., Peelle, J., Mahon, B., Buxbaum, L., Majid, A., Brysbaert, M., et al. (2023). What we mean when we say semantic: A consensus statement on the nomenclature of semantic memory. *OSF preprint*.

Rohrbach, S., Šiaučiulis, M., Chisholm, G., Pirvan, P.-A., Saleeb, M., Mehr, S. H. M., Trushina, E., Leonov, A. I., Keenan, G., Khan, A., et al. (2022). Digitization and validation of a chemical synthesis literature database in the chempu. *Science*, 377(6602):172–180.

Seifrid, M., Pollice, R., Aguilar-Granda, A., Morgan Chan, Z., Hotta, K., Ser, C. T., Vestfrid, J., Wu, T. C., and Aspuru-Guzik, A. (2022). Autonomous chemical experiments: Challenges and perspectives on establishing a self-driving lab. *Accounts of Chemical Research*, 55(17):2454–2466.

Shi, Y.-Z., Hou, H., Bi, Z., Meng, F., Wei, X., Ruan, L., and Wang, Q. (2024a). AutoDSL: Automated domain-specific language design for structural representation of procedures with constraints. In *Annual Meeting of the Association for Computational Linguistics*.

Shi, Y.-Z., Li, H., Ruan, L., and Qu, H. (2024b). Constraint representation towards precise data-driven storytelling. In *IEEE Visualization and Visual Analytics Gen4DS*.

Shi, Y.-Z., Li, S., Niu, X., Xu, Q., Liu, J., Xu, Y., Gu, S., He, B., Li, X., Zhao, X., et al. (2023a). PersLEARN: Research training through the lens of perspective cultivation. In *Annual Meeting of the Association for Computational Linguistics*.

Shi, Y.-Z., Xu, M., Hopcroft, J. E., He, K., Tenenbaum, J. B., Zhu, S.-C., Wu, Y. N., Han, W., and Zhu, Y. (2023b). On the complexity of Bayesian generalization. In *International Conference on Machine Learning*.

Shi, Y.-Z., Xu, Q., Meng, F., Ruan, L., and Wang, Q. (2024c). Abstract Hardware Grounding towards the Automated Design of Automation Systems. In *International Conference on Intelligent Robotics and Applications*.

Singh, S. H., van Breugel, F., Rao, R. P., and Brunton, B. W. (2023). Emergent behaviour and neural dynamics in artificial agents tracking odour plumes. *Nature Machine Intelligence*, 5(1):58–70.

Steiner, S., Wolf, J., Glatzel, S., Andreou, A., Granda, J. M., Keenan, G., Hinkley, T., Aragon-Camarasa, G., Kitson, P. J., Angelone, D., et al. (2019). Organic synthesis in a modular robotic system driven by a chemical programming language. *Science*, 363(6423):eaav2211.

Strateos (2023). Autoprotocol specification. https://autoprotocol.org/specification/.

Szymanski, N. J., Rendy, B., Fei, Y., Kumar, R. E., He, T., Milsted, D., McDermott, M. J., Gallant, M., Cubuk, E. D., Merchant, A., et al. (2023). An autonomous laboratory for the accelerated synthesis of novel materials. *Nature*, 624(7990):86–91.

Vaucher, A. C., Zipoli, F., Geluykens, J., Nair, V. H., Schwaller, P., and Laino, T. (2020). Automated extraction of chemical synthesis actions from experimental procedures. *Nature Communications*, 11(1):3601.

Wang, B., Wang, Z., Wang, X., Cao, Y., A Saurous, R., and Kim, Y. (2023a). Grammar prompting for domain-specific language generation with large language models. In *Advances in Neural Information Processing Systems*.

Wang, H., Fu, T., Du, Y., Gao, W., Huang, K., Liu, Z., Chandak, P., Liu, S., Van Katwyk, P., Deac, A., et al. (2023b). Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60.

Wang, Z., Cruse, K., Fei, Y., Chia, A., Zeng, Y., Huo, H., He, T., Deng, B., Kononova, O., and Ceder, G. (2022). ULSA: Unified language of synthesis actions for the representation of inorganic synthesis protocols. *Digital Discovery*, 1(3):313–324.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2021). Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Xie, T., Li, Q., Zhang, Y., Liu, Z., and Wang, H. (2024). Self-improving for zero-shot named entity recognition with large language models. *arXiv preprint arXiv:2311.08921*.

Zhang, J., Chen, B., Zhang, L., Ke, X., and Ding, H. (2021). Neural, symbolic and neural-symbolic reasoning on knowledge graphs. *AI Open*, 2:14–35.

A Additional remarks

A.1 Rationale for the evaluation metrics

Direct comparisons across entire sentences under BLEU and ROUGE scores would indeed pose a problem — it may be problematic considering instructions that look similar could have very different semantics. Therefore, to circumvent this issue, we convert all results into a standardized JSON-style format for data representation, and comparisons are made between key-value pairs rather than entire sentences, effectively resolving the metric concern.

Let us consider comparing similarity between the following two instructions *"... pour hot water ..."* and *"... pour cold water ..."*. We transform them into the following JSON-style format:

```
{
    ...
    function_name: "pour",
    reagent: "water",
    temperature: "hot",
    ...
}
...
{
    ...
    function_name: "pour",
    reagent: "water",
    temperature: "cold",
    ...
}
```

The comparison between the two sentences is then transformed into a comparison between two JSON code blocks. We calculate the similarity score cumulatively based on the similarity between the values of matched pairs of keys. For instance, for the key "temperature", the values "hot" and "cold" yield a low similarity score under the ROUGE, BLEU, and even the Exact Match metrics. As "temperature" is one of the major keys within configuration parameters, a high penalty in this dimension significantly affects the cumulative similarity score. With this fine-grained comparison metric, we can comprehensively track the distinctions and commonalities between results without losing expressivity regarding the quantities.

We also acknowledge that there are advanced evaluation metrics, especially in the recent works where LLMs are leveraged as external judges and achieve considerable performance in general testing cases. Our choice of *less advanced* metrics is driven by the intention to focus specifically on domain-specific knowledge, which constitutes the primary scope of this paper and may be relatively sparse in general LLMs. Nonetheless, the exploration of more sophisticated evaluation metrics represents a promising avenue for future research.

A.2 Insight behind the design of PDG

We have constructed the operation dependence graph on the syntax level and the reagent flow graph on the semantics level. Indeed, the two analytical results come with a *duality*. In the operation dependence graph, vertices represent operations and edges represent reagents passed between them. In contrast, the reagent flow graph uses vertices for reagent states and edges for operations causing state transitions. Interestingly, the vertices of one of the two graphs can be one-to-one mapped to the edges of another, echoing the duality. On a higher level, we say that the former provides an *experimenter-centered view* while the latter offers a *reagent-centered view*. These two perspectives are complementary on encoding both the information of the interventions to the environment and the status of the environment itself. Consequently, by leveraging such duality, we are able to track spatial dynamics, *e.g.*, the variance of required resources, and temporal dynamics, *e.g.*, the context of sequential operations, simultaneously on the PDG.

A.3 Computational complexity of the framework

Let us consider a new coming protocol with k steps, with each step configured by a constant number of parameters, denoted as c. On the syntax level, the primary computation bottleneck arises during DSL program synthesis, where the EM Algorithm exhibits a worst-case complexity of $O(c^k)$. This is a highly conservative estimate, as mainstream optimization approaches can solve the EM much more efficiently. On the semantics level, the bottleneck occurs during reagent flow analysis, which consumes $O(k^2)$ complexity. Notably, only approximately 10% of the steps are included in the nested loop for reagent flow construction, as about 90% of the steps are linearly connected. On the execution level, the protocol execution model also exhibits $O(k^2)$ complexity, encompassing both forward and backward tracing. This can be optimized by replacing the full tracing strategy with a sliding window built upon the topological dependencies between steps. Although the complexities of the algorithms at these three levels are tractable, there is substantial room for improving the efficiency of the framework. Investigating methods to speed up the translation process for protocols with extremely high complexity would be a valuable area of research.

A.4 Generality of the framework

The general applicability of our proposed framework beyond experimental sciences can indeed be a common concern. The core value of translating NL-based protocols into formats suitable for machine execution substantially lies in facilitating experiments in self-driving laboratories, thereby accelerating scientific discovery. Experimental protocols come with unique properties and challenges, such as the fine-grained incorporation of domain-specific knowledge, the non-trivial dependency topologies between operations, the long-horizon lifecycles of intermediate productions, and the necessity for precise execution without run-time errors. These factors shape the scope of our research problem, emphasizing the need to handle protocols with stringent terminology and formatting.

Despite the specific scope of this paper, we are open to exploring the potential for generalizing our framework to other domains with similar properties and challenges as those found in scientific experiments — such as cooking. Imagine a self-driving kitchen that automatically prepares all ingredients and executes all procedures for cooking a meal according to NL-based recipes. Such self-driving kitchens would also benefit significantly from translating human-oriented recipes into formats suitable for machine execution. In the following, we present a running example of such a translation, adapted from a use case of the Corel[1] DSL.

The protocol after pre-processing is as follows.

```
Pasta Bolognese

Yield: 2 plates

Ingredients:

- 8 [ounces] white fresh {pasta}

- 1 [floz] olive {oil}

- 1/4 [ounce] {garlic}; minced

- 4 [ounces] {onions}; chopped

- 4 [ounces] shallow fried {beef}; minced

- 1 - 1 1/2 [ounce] lean prepared {bacon}

- 1/3 [cup] red {wine}
```

[1] Visit `https://fse.studenttheses.ub.rug.nl/25731/` for documentation.

```
20
21 - 150 [gram] raw {carrots}; thinly sliced
22
23 - 2/3 [ounce] concentrated {tomato puree}
24
25 - 4 [ounces] red {sweet pepper}; cut julienne
26
27 - 1 [ounce] {parmesan} cheese
28
29 Instructions:
30
31 Add the @oil@ to a large saucepan, heat to <300 F>, and saute the
       @onions@.
32
33 After |2 minutes|, add the @garlic@. Keep on medium to high heat, and don
       't stir.
34
35 After |2 minutes| more, add the @beef@.
36
37 Fry the @bacon@ in a separate pan, on high heat. Remove liquified fat
       when done.
38
39 Boil @pasta@ in a medium pan, until al dente (~|8 minutes|). Drain when
       done.
40
41 Once the @beef@ is done, add the @carrots@, @sweet pepper@ and @tomato
       puree@.
42
43 Slowly add the @wine@ as well, to not lower the temperature. Let it
       simmer (but not boil) for |5-10 minutes|.
```

Given the protocol as the input of our framework, the resulting DSL program is as follows.

```
1 add(slot = "oil", target = "large saucepan", container = plate_1, emit =
      mixture_1);
2
3 heat(target = mixture_1, temperature = 300F, container = plate_1, postcon
       = stop());
4
5 saute(target = "onions", container = plate_2, duration = 2mins);
6
7 add(slot = "garlic", target = mixture_1, container = plate_1, emit =
      mixture_2);
8
9 heat(target = mixture_2, temperature = 325F, container = plate_1,
      duration = 2mins);
10
11 add(slot = "beef", target = mixture_2, container = plate_1, emit =
      mixture_3);
12
13 heat(target = mixture_2, temperature = 325F, container = plate_1,
      postcond = check_done(target = "beef"));
```

```
14
15 fry(target = "bacon", temperature = 350F, container = pan_1, postcond =
       remove(target = "liquified fat"));
16
17 boil(target = "pasta", temperature = 212F, container = pan_2, duration =
       8mins, postcond = drain());
18
19 add(precond = check_done(target = "beef"), slot = ["carrots", "sweet
       pepper", "tomato puree"], target = mixture_3, container = plate_1,
       emit = mixture_4);
20
21 add(slot = "wine", target = mixture_4, container = plate_1, pace = 1mL/s)
       ;
22
23 simmer(target = mixture_4, temperature = 211F, duration = 7.5mins);
```

In this example, we observe that the NL-based recipe possesses ambiguities and omissions. Our translation framework addresses these challenges by structuring the recipe on the syntax level, completing the latent information on the semantics level, and linking the programs with necessary resources, such as the usage of plates, on the execution level. Due to the modularity of DSLs, although Corel's distribution of syntactic and semantic features differs significantly from those of DSLs used for representing experimental protocols, our translator can generalize to this new target domain through the structure of rules, namely *rule-based generalization* (Shi et al., 2023b).

A.5 The motivations behind this work

In this work, we study the problem of translating experimental protocols designed for human experimenters into formats suitable for machine execution. Our primary motivation is to bridge the existing gap between machine learning algorithms in the field of AI for science, such as molecular design, and the grounded experimental verification facilitated by self-driving laboratories. Conventional workflows for setting up self-driving laboratories and conducting physical experiments necessitate deep integration with domain experts, significantly impeding the progress of machine learning researchers in verifying and iterating their findings. Consequently, our framework aims to provide an infrastructure that enables these researchers to advance their machine learning algorithms and seamlessly validate their findings, thereby closing the loop of automatic scientific discovery.

To meet the requirements of such infrastructure, we conduct a systematic study to identify existing gaps in protocol translation between human experimenters and automatic translators in self-driving laboratories. From the study, we derive design principles that emulate human cognitive processes involved in protocol translation. Under the guidance of these design principles, we develop the three-stage framework that integrates cognitive insights from human experts with approaches from program synthesis, automaton construction, and counterfactual analysis. On the syntax level, we synthesize the operation dependence graph to transform NL-based protocols into structured representations, thereby making explicit the operation-condition mappings and the control flows. On the semantics level, we analyze the reagent flow graph to reconstruct the complete lifecycles of intermediate products, addressing the latent, missing, or omitted properties and values. On the execution level, we contextualize both the operation dependence graph and the reagent flow graph within spatial and temporal dynamics, resulting in the protocol dependence graph. This graph conducts counterfactual reasoning to detect potential conflicts or shortages of execution resources and to identify inappropriate combinations of operations in execution sequences.

B Ethics statement

B.1 Human participants

The meta-evaluation included in this work has been approved by the Institutional Review Board (IRB) of Peking University. We have been committed to upholding the highest ethical standards in

conducting this study and ensuring the protection of the rights and welfare of all participants. We paid the domain experts a wage of $22.5/h, which is significantly higher than the standard wage.

We have obtained informed consent from all participants, including clear and comprehensive information about the purpose of the study, the procedures involved, the risks and benefits, and the right to withdraw at any time without penalty. Participants were also assured of the confidentiality of their information. Any personal data collected (including name, age, and gender) was handled in accordance with applicable laws and regulations.

B.2 Corpora collection

We carefully ensure that all experimental protocols incorporated into our corpora strictly adhere to open access policies, governed by the Creative Commons license. This approach guarantees full compliance with copyright and intellectual property laws, eliminating any potential infringement or unauthorized use of protected materials. By exclusively utilizing resources that are freely available and legally distributable, we uphold the highest standards of ethical conduct in research, fostering an environment of transparency and respect for the intellectual property rights of others. This commitment ensures that our work not only advances the frontiers of knowledge but does so in a manner that is both legally sound and ethically responsible.

C Implementation details

C.1 Details of pre-processing

We employ the SpaCy Dependency Parser to analyze the syntactic structure of protocol $\mathbf{c}$, which allows for the extraction of verbs and the identification of associated objects and modifiers (Honnibal and Johnson, 2015). After parsing, these verbs are aligned with corresponding operational actions $o \in T_{\text{op}}$ in the DSLs by maximizing the cosine similarity between their word2vec representations and those of the DSL operations. Furthermore, we utilize an advanced few-shot Named Entity Recognition (NER) algorithm, based on large language models, to accurately identify and classify entities $\mathbf{e}_t \in \mathcal{E}$ within the text (Xie et al., 2024). The rationale of integrating LLMs with classical parsing techniques lies in leveraging the advanced natural language processing capabilities of LLMs while mitigating their inherent uncertainties.

The prompt for NER is as follows.

```
Given entity label set: {label_set}.
Please name the entities in the given text. Based on the given entity
    label set, provide answer in the following JSON format: [{"Entity
    Name": "Entity Label"}]. If there is no entity in the text, return
    the following empty list: [].
Please note that entities have already been annotated with [], no need to
     extract and analyze other entities.
{cases}
Text: {query}
Answer:
```

C.2 Details of reagent flow analysis

We extract reagents from the natural language descriptions of protocols using two utility functions, KILLS and DEFINES. KILLS identifies the reagent consumed in an operation, while DEFINES identifies the reagent introduced in an operation. Due to the potential for a single chemical substance to have multiple names among other factors, it is impractical to rely solely on string matching to determine if a reagent is killed in a given operation. Instead, we employ a method based on prompt engineering with LLMs for this analysis.

The following prompt is used to analyze whether the input of the current operation is the output of a previous operation:

```
This instruction describes a step in an experimental process, which includes one action, multiple parameters, and one output.
Please help analyze the output of this instruction. I will provide a list of potential outputs. You need to assist in determining which of these outputs is most suitable for this instruction.
Note that you must choose one output from the list. Please output only a string without any explanation.

[Examples]
Instruction: {"action": "add", "reagent": ["glycoblue"], "output": ""}
Potential output list: "RNA", "mRNA"
Output: "RNA"

Instruction: {"action": "add", "concentration": ["1:10 volume 5 M NaCl"], "output": ""}
Potential output list: "a μMACS column", "solution"
Output: "solution"

Instruction: {"action": "heat", "reagent": ["limestone"], "output": ""}
Potential output list: "water", "NaCl"
Output:

[Question]
Instruction: {Instruction}
Potential output list: {Input}
Output:
```

Additionally, this prompt is used to determine if reagents in the current memory $M(\Gamma)$ are killed by the current operation:

```
This instruction describes a step in an experimental process, which includes one action, multiple parameters-including various reagents-and one output.
Please help analyze the missing reagents of this instruction. I will provide a list of potential reagents. You need to help me analyze which of these reagents might be the ones omitted from the current instruction.
Please note how many reagent parameters are missing from the current instruction. It is possible that some reagent parameters cannot be completed with the list provided. Please output only a comma-separated list of strings without any explanation.

[Examples]
Instruction: {"action": "add", "reagent": [""], "output": ""}
Potential reagent list: "RNA", "glycoblue"
Reagents: "glycoblue"

Instruction: {"action": "add", "concentration": ["1:10 volume"], "reagent": ["", ""], "output": ""}
Potential reagent list: "μMACS", "solution", "NaCl"
Reagents: "NaCl", "μMACS"

```

```
Instruction: {"action": "use", "reagent": ["BamHI", "XhoI", ""], "device": ["PCR amplification"], "output": ""}
Potential reagent list: "agar", "food"
Reagents:

[Question]
Instruction: {Instruction}
Potential reagent list: {Memory}
Reagents:
```

In the process of synthesizing operation dependencies on the syntax level, we implement the DEFINES function. This involves pattern matching after pre-processing to accurately define reagents in each operation.

C.3 Cost of the implementation

The computational cost of our algorithm primarily arises from the expenses associated with API calls to LLMs. We selected OpenAI's `gpt-3.5-turbo-0125` model for our experiments. Across 75 test protocols, we executed 1816 queries to achieve syntax-level translation, resulting in structured protocols. At the semantic level, we conducted 4062 queries for completion tasks (including translating protocols retrieved from training dataset). During these experiments, the cost model charged US$0.50 per million tokens for inputs and US$1.50 per million tokens for outputs. Consequently, our expenditures were approximately US$10. Additionally, we utilized OpenAI's `text-embedding-ada-002` model to embed the training dataset and build a vector database, which incurred a cost of about US$7.

D The testing set

D.1 Collection

The real experiments for the testing set are retrieved from open-sourced websites run by top-tier publishers, including Nature's Protocolexchange[2], Cell's Star-protocols[3], Bio-protocol[4], Wiley's Current Protocols[5], and Jove[6].

D.2 Showcases

```
[Protocol 1 - Biochemistry]
Preparation of lysates
1. Harvest approximately 1 x 10^7 cells by centrifugation at 2000 RPM for 5 min. Aspirate media and resuspend cell pellet with 1 mL of ice-cold PBS and transfer to a 1 mL centrifuge tube. Microcentrifuge at 2000 RPM for 5 min at 4 °C.
2. Aspirate PBS, and then add Hypotonic Buffer (supplemented with 1% Triton X-100, to disrupt membrane and cytoskeleton-bound MEKK1 fractions).
3. Cell lysates are homogenized by passing through 22-gauge needles, and tubes are put on ice for 15 min to complete the lysis. Crude extracts are then centrifuged at 2500 RPM for 5 min. Supernatants are transferred to fresh centrifuge tubes, and cold 5 M NaCl is added to
```

[2] https://protocolexchange.researchsquare.com/
[3] https://star-protocols.cell.com/
[4] https://bio-protocol.org/en
[5] https://currentprotocols.onlinelibrary.wiley.com/
[6] https://www.jove.com/

each sample to make a salt concentration of between 0.7–1.0 M to disrupt protein-protein interactions.
4. Spin the crude extracts by ultracentrifugation at 55000 RPM to properly pellet residual insoluble proteins from the extract. Transfer supernatants into fresh centrifuge tubes.
Immunoprecipitation
5. Rinse Protein A beads in Hypotonic Buffer and place on ice until ready for use.
6. Take a volume of cell lysates (prepared as described above), and dilute with Hypotonic Buffer to 250–500 mM salt to enable protein-protein interactions.
7. Add 2 μg of preclearing antibody to the diluted lysate (e.g., anti-Myc or anti-p65), vortex, add 50 μL of Protein A beads, and rock for 45 min.
8. Touchspin samples, and transfer supernatant to a fresh tube.
9. Add 2 μg of polyclonal anti-MEKK1 to the lysates, and rock for 1 h. After this period, add 50 μL of Protein A beads and rock tubes at 4 $^{\circ}$C for 1 h.
10. Touchspin beads, wash beads with hypotonic buffer (supplemented with NaCl to a concentration of 300 mM), vortex, and rock for 10 min. In total, 3–5 washes of the beads are performed.
11. Finally, wash once with Hypotonic Buffer, and resuspend in Kinase Assay Buffer. Purified MEKK1 may be stored by snap-freezing in liquid nitrogen and long-term storage at -80 $^{\circ}$C. Kinase assay Following preparation of MEKK1 immunoprecipitates (as above), incubate with 7 μg of JNKK1(K131M) along with 5 μCi of ATP in Kinase Assay Buffer for 30 min at 30 $^{\circ}$C."

[Protocol 2 - Genetics]
1. Note that everything is in DEPC water. Inoculate W303a cells expressing different TOR1-RR variants in 2 mL SC medium overnight.
2. Subculture the cells starting from OD600=0.1 in 10 mL SC media, shake vigorously at 30 $^{\circ}$C, 300 RPM for around 4–6 h until OD600=0.4–0.5.
3. Collect the cells by spinning down without freezing on ice. Discard supernatant.
4. Re-suspend cells with 1 mL water and transfer to a 1.5 eppendorf tube, quickly spin down at 3,000 x g for 15 sec.
5. Re-suspend cell pellet in 400 μL of AE buffer at room temperature.
6. Add 40 μL 10% SDS (final around 1%) and vortex briefly at room temperature (RT).
7. Immediately add 500 μL hot phenol/AE (put in 65 $^{\circ}$C for 10 min before use), vortex vigorously for 1 min. Incubate at 65 $^{\circ}$C for 5 min. Briefly vortex every 30 sec.
8. Immediately freeze by dumping into liquid nitrogen. Wait to thaw at RT (put in 30 $^{\circ}$C to thaw may crack the tube).
9. Centrifuge for 10 min on a standard laboratory microfuge at 20,000 x g at RT.
10. Transfer around 400 μL supernatant to a new eppendorf tube. Recycle the lower phenol fraction carefully following the chemical safety protocol in your laboratory.

11. Add equal volume (400 μL) phenol: CHCl3/AE-Na. Vortex vigorously for 1 min at RT.
12. Spin down at 20,000 x g for 5 min in a standard laboratory microfuge.
13. Transfer supernatant (around 350 μL) to a fresh 1.5 mL eppendorf tube. Add CHCl3: isoamyl alcohol (24:1). Vortex vigorously for 1 min at RT.
14. Transfer aqueous supernatant to a fresh 1.5 mL microfuge tube. If white cloudy precipitate is observed between the aqueous phase and organic phase, repeat steps 17-18.
15. Add 1/10 volume of 3 M NaOAc (pH 5) and vortex vigorously. Add 2.5 volumes of ethanol. Vortex again.
16. Place at -20 $^{\circ}$C for at least 30 min.
17. Spin down in the microfuge at 20,000 x g, 15 min at 4 $^{\circ}$C. RNA pellet is usually visible.
18. Add ice-cold 75% EtOH, place at 4 $^{\circ}$C for around 10 min. Vortex and spin down on microfuge 20,000 x g, 15 min at 4 $^{\circ}$C. Discard supernatant. Suck out the liquid droplets in the tube. The white RNA pellet will turn clear when it dries out. Add 30-50 μL ddH2O (DEPC) immediately after it becomes clear. Do not let the RNA over-dry, which will make it difficult to dissolve. If RNA pellet is over-dry, dissolve RNA at 37 $^{\circ}$C for 30 min. Store RNAs at -80 $^{\circ}$C for more than 2 months."

[Protocol 3 - Biomedical]
1. Passage through a 45 μm filter. Add 100 μL/well of 100 μg/mL salmon sperm DNA to a 96-well Microtest assay plate.
2. Wrap the plate with plastic wrap and incubate at 4 $^{\circ}$C overnight.
3. Discard the coating antibody solution and wash the plate with 1x PBS-Tween 6 times.
4. Dry the plate and add 100 μL of blocking solution per well to the plate.
5. Incubate the plate at room temperature (RT) for 1.5 h.
6. Discard the blocking solution and wash the plate with 1x PBS-Tween 5 times.
7. Dry the plate and keep it at 4 $^{\circ}$C for later use.
8. Harvest the spleen and create a single-cell suspension by gently smashing spleen pieces with the frosted surface of a pair of microscope slides in 5 mL of DMEM.
9. Transfer the cells into 50 mL conical tubes and spin down the cells at 300 RCF for 5 min at 4 $^{\circ}$C.
10. Discard the supernatant with aspiration without disturbing the pellet.
11. Re-suspend the cells with 5 mL of 0.17 M ammonium chloride and keep the cells on ice for 5 min.
12. Add 15 mL DMEM to the cells and spin at 300 RCF for 5 min at 4 $^{\circ}$C.
13. Discard the supernatant and re-suspend the cells with 20 mL of DMEM and count the cells.
14. Re-suspend 2 x 10^7 cells in 2 mL of 10% DMEM and make a three-fold serial dilution (a total of 8 dilutions) with 10% DMEM.
15. Add 50 μL/well of the serial dilutions on the DNA-coated plate and centrifuge at 300 RCF for 5 min at 4 $^{\circ}$C.

16. Incubate the cells at 30 °C for 2 h in a cell-culture incubator with 6% CO2.
17. Add 50 μL/well of biotin-conjugated anti-IgM or anti-IgG (1:350 in 10% DMEM) to the cells.
18. Centrifuge the cells at 300 RCF for 5 min at 4 °C and incubate the cells overnight in a cell-culture incubator with 6% CO2.
19. Discard the cells and wash the plates 10 times with 10x PBS-Tween 20.
20. Dry the plates and add 50 μL of streptavidin alkaline phosphatase (1:1,000 in 1% BSA/PBS) to the plate.
21. Incubate the plate at RT for 1 h and wash the plate 10 times with 10x PBS-Tween 20.
22. Dry the plate and add 50 μL/well of 1 mg/mL BCIP in AMP buffer to develop the plate.
23. When the spots are clearly visible under a dissecting microscope, stop the development by discarding the BCIP solution and rinsing the plate with tap water thoroughly.
24. Spots can be counted using a dissecting microscope or using an ELISpot reader."

D.3 Instruction for human experts

```
Instruction for Human Study on Protocol Translation and Parameter Completion
[Objective]
The purpose of this study is to evaluate the accuracy and completeness of translating natural language laboratory protocols into a structured JSON representation and to assess the manual completion of missing parameters within these protocols.
[Experimental Tasks]
Participants in this study will perform two main tasks:
1. Translation of Natural Language Protocols to JSON-Structured Representation
2. Manual Parameter Completion in JSON-Structured Protocols
[Task 1: Translation of Natural Language Protocols to JSON-Structured Representation]
[Description]
Participants will be provided with a set of laboratory protocols written in natural language. The task is to translate each protocol into a JSON-structured format. This involves accurately mapping the operations, input reagents, and conditions specified in the natural language description to a precise JSON schema.
[Procedure]
1. Read the provided natural language protocol carefully.
2. Identify and extract the key elements of the protocol, including: Operations (e.g., dissolve, mix, heat)/Input reagents (e.g., sodium chloride, distilled water)/Conditions (e.g., temperature, time, concentration)
3. Construct a JSON representation that clearly reflects the structure and content of the protocol. Ensure that each element is correctly mapped to its corresponding key and value pairs.
[Example]
```

```
Extract total RNA from at least 2 x 10^6 cells using TRIZOL reagent.
{"action": "extract", "output": "total RNA", "reagent": ["TRIZOL reagent
    "], "volume": ["at least 2 x 10^6 cells"], "container": [""]}
[Manual Parameter Completion in JSON-Structured Protocols]
[Description]
Participants will receive a set of JSON-structured protocols with certain
     parameters intentionally left incomplete. The task is to manually
    complete these parameters based on domain knowledge and logical
    inference.
[Procedure]
1. Review the provided JSON-structured protocol.
2. Identify any missing or incomplete parameters.
3. Use your expertise to infer the missing information. This may include:
     Estimating reasonable values for missing quantities or conditions;
Ensuring consistency and coherence within the protocol.
4. Complete the JSON structure with the inferred parameters, maintaining
    accuracy and logical consistency.
[Example]
{"action": "apply", "output": "known DHB cluster signals", "device":
    ["<<<MASK>>>"]}
{"action": "apply", "output": "known DHB cluster signals", "device":
    ["<<<a pneumatic sprayer system>>>"]}
```

E Case studies

E.1 Contributions of the components in our translator

We provide a series of case studies to illustrate the distinctions between the behaviors of the components within our proposed framework and those of the baselines qualitatively in Tab. A1.

Table A1: **Distinctions between the behaviors of the three components within our proposed framework**

Original Text	Stage 1	Feature	Stage 2	Feature	Stage 3	Feature
Kill most the contaminating spores that have germinated. Centrifuge the spore mixture at <MASK> for 5 min.	ELIMINATE: [[Reg: contaminating spores]]; SPIN: [[Reg: spore mixture], [Time: 5min], [Force: MASK]]	IN: [[contaminating spores], [spore mixture]]	ELIMINATE: [[Reg: contaminating spores]]; SPIN: [[Reg: spore mixture], [Time: 5min], [Force: 1,200g]]	Latent semantics of unknown unknowns (force)	ELIMINATE: [[Reg: contaminating spores]] -> spore mixture; SPIN: [[Reg: spore mixture], [Time: 5min], [Force: 1,200g]]	Reg: spore mixture; No specific volume provided
Add pre-hybr soln directly to the hybridization reaction to get hybrid molecule. Incubate for <MASK>.	ADD: [[Reg: pre-hybr soln], [Reg: hybridization reaction]] -> hybrid molecule; INCUBATE: [[Time: MASK]]	IN: [[pre-hybr soln], [hybridization reaction]] -> hybrid molecule	ADD: [[Reg: pre-hybr soln], [Reg: hybridization reaction]] -> hybrid molecule; INCUBATE: [[Time: 10min]]	Latent semantics of known unknowns (time)	ADD: [[Reg: pre-hybr soln], [Reg: hybridization reaction]] -> hybrid molecule; INCUBATE: [[Reg: hybrid molecule], [Time: 10min]]	Reg: pre-hybr soln, hybridization reaction; No specific volume provided
Confirm positive colonies by PCR. Take fluorescence images under <MASK>.	CONFIRM: [[Device: PCR]] -> positive colonies; TAKE: [[Device: MASK]] -> fluorescence images	IN: [PCR]] -> positive colonies; IN: [[positive colonies]] -> fluorescence images	CONFIRM: [[Device: PCR]] -> positive colonies; TAKE: [[Device: microscope]] -> fluorescence images	Latent semantics of unknown unknowns (device)	CONFIRM: [[Device: PCR], [Reg: RNAs]] -> positive colonies; TAKE: [[Device: microscope]] -> fluorescence images	Reg: RNAs; No specific volume provided

E.2 Running cases of our translator handling specific challenges

Ensuring the safety and correctness of translated protocols is an exceptionally challenging task. Several factors contribute to these challenges, including accurately mapping operations to their corresponding configuration parameters (Tab. A2), precisely parsing control flows from natural language (Tab. A3), completing latent semantics with domain-specific knowledge (Tab. A4), inferring missing or omitted key information (Tab. A5), tracking resource capacities (Tab. A6), and verifying the safety of run-time execution of experiments (Tab. A7). Consequently, we have made specific efforts in response to these challenges, resulting in our design of translator. Here we provide several running examples to demonstrate our translator's capability on handling the challenging factors respectively.

Table A2: **Running cases on the syntax level regarding operation-condition mapping**

Original Text	Syntax Level	Action	Conditions
Spin media at 500-1,000 x g for 10 min (optional), pre-x g for 10 min, filter with 0.22 um PES membrane, freeze at -80°C.	SPIN: [[Speed: 500-1,000 x g], [Time: 10 min]] -> filtered media, FILTER: [[Device: 0.22 um PES membrane]] -> filtered media, FREEZE: [[Temperature: -80°C]] -> frozen media	Spin, Filter, Freeze	Speed: 500-1,000 x g, Time: 10 min, Device: 0.22 um PES membrane, Temperature: -80°C
Thaw 4 ml supernatant on ice, add 4 ml XBP buffer.	THAW: [[Volume: 4 ml], [Reagent: supernatant]] -> thawed supernatant, ADD: [[Reagent: XBP buffer], [Volume: 4 ml]] -> sample/XBP mix	Thaw, Add	Volume: 4 ml, Temperature: On ice
Add sample/XBP mix to exoEasy maxi spin column, centrifuge 1-3 min at 500 x g, discard flow-through.	ADD: [[Reagent: sample/XBP mix], [Container: spin column]] -> flow-through, CENTRIFUGE: [[Speed: 500 x g], [Time: 1-3 min]] -> flow-through, DISCARD: [[Reagent: flow-through]] ->	Add, Centrifuge, Discard	Container: Spin column, Speed: 500 x g, Time: 1-3 min
Add 10 ml XWP to spin column, centrifuge 5 min at 5,000 x g, transfer column to fresh collection tube.	ADD: [[Reagent: XWP], [Volume: 10 ml]] -> , CENTRIFUGE: [[Speed: 5,000 x g], [Time: 5 min], [Container: spin column]] -> , TRANSFER: [[Container: fresh collection tube]] ->	Add, Centrifuge, Transfer	Volume: 10 ml, Speed: 5,000 x g, Time: 5 min, Container: Spin column, Fresh collection tube
Add 700 uL Qiazol to spin column, centrifuge 5 min at 5,000 x g, spin PLG tubes 30 s at 16,000 x g.	ADD: [[Reagent: Qiazol], [Volume: 700 uL]] -> , CENTRIFUGE: [[Speed: 5,000 x g], [Time: 5 min]] -> , SPIN: [[Speed: 16,000 x g], [Time: 30 s], [Container: PLG tubes]] ->	Add, Centrifuge, Spin	Volume: 700 uL, Speed: 5,000 x g, Time: 5 min, Speed: 16,000 x g, Time: 30 s, Container: Spin column, PLG tubes
Add flow-through to PLG tube, vortex 5 s, incubate 5 min at RT.	ADD: [[Reagent: flow-through], [Container: PLG tube]] -> , VORTEX: [[Time: 5 s]] -> , INCUBATE: [[Time: 5 min], [Temperature: RT]] ->	Add, Vortex, Incubate	Container: PLG tube, Time: 5 s, Time: 5 min, Temperature: RT
Add 90 uL chloroform.	ADD: [[Volume: 90 uL], [Reagent: chloroform]] ->	Add	Volume: 90 uL
Shake vigorously for 15 s, incubate 2-3 min at RT.	SHAKE: [[Time: 15 s]] -> , INCUBATE: [[Time: 2-3 min], [Temperature: RT]] ->	Shake, Incubate	Time: 15 s, Time: 2-3 min, Temperature: RT
Centrifuge 15 min at 12,000 x g, transfer upper aqueous phase to new tube.	CENTRIFUGE: [[Speed: 12,000 x g], [Time: 15 min]] -> upper aqueous phase, TRANSFER: [[Container: new tube]] -> upper aqueous phase	Centrifuge, Transfer	Speed: 12,000 x g, Time: 15 min, Container: New tube
Add 2 volumes 100 ethanol, mix.	ADD: [[Volume: 2 volumes], [Reagent: 100 ethanol]] -> ethanol mixture, MIX: [[Reagent: ethanol mixture]] ->	Add, Mix	Volume: 2 volumes

Table A3: **Running cases on the syntax level regarding operation control flows**

Original Text	Syntax Level	Action	Control Flows
Centrifuge the cell suspension at 200 x g at room temperature for 5 min.	`CENTRIFUGE: [[Force: 200 x g], [Temperature: room temperature], [Time: 5 min]] -> cell pellet`	Centrifuge	Linear
Remove the supernatant.	`REMOVE: [[Output: supernatant]] ->`	Remove	Linear
Suspend the cell pellet with 2 ml ACK lysing buffer for 1 min to deplete red blood cells.	`SUSPEND: [[Volume: 2 ml], [Reagent: ACK lysing buffer], [Time: 1 min]] -> depleted cell suspension`	Suspend	Linear
If red blood cells are not completely depleted, repeat the ACK lysing buffer step until they are.	`REPEAT: [[Reagent: ACK lysing buffer], [Condition: if red blood cells are not completely depleted]] ->`	Repeat	Non-linear
Filter the cell suspension through a 40 um nylon strainer.	`FILTER: [[Container: 40 um nylon strainer]] -> filtered cell suspension`	Filter	Linear
Wash the strainer with 2 ml 1x DPBS for 5 min.	`WASH: [[Container: strainer], [Reagent: 1x DPBS], [Volume: 2 ml], [Time: 5 min]] ->`	Wash	Linear
Wash the cell pellet with 1x DPBS with 20 ng/ml murine M-CSF in a 100 mm Petri dish.	`WASH: [[Reagent: 1x DPBS with 20 ng/ml murine M-CSF], [Container: 100 mm Petri dish]] ->`	Wash	Linear
Suspend in 15 ml complete DMEM medium.	`SUSPEND: [[Volume: 15 ml], [Reagent: complete DMEM medium]] -> cell suspension in DMEM`	Suspend	Linear
Incubate at 37 °C, 5 CO2.	`INCUBATE: [[Temperature: 37 °C], [Environment: 5 CO2]] -> incubated cells`	Incubate	Linear
After 3 days, replace half of the medium with fresh complete DMEM medium.	`REPLACE: [[Reagent: fresh complete DMEM medium], [Time: after 3 days]] ->`	Replace	Linear
Repeat this step every 2 days.	`REPEAT: [[Reagent: fresh complete DMEM medium], [Frequency: every 2 days]] ->`	Repeat	Non-linear

Table A4: **Running cases on the semantics level regarding known unknowns**

Original Text	Semantic Level	Known Unknowns
Transfer the sample (plasma, cell suspension) into a glass centrifuge vial.	`TRANSFER: [[Reagent: the sample (plasma, cell suspension)], [Container: a glass centrifuge vial]] ->`	
Adjust the volume to 1 ml with PBS.	`MODIFY: [[Output: heparinized blood.1 ml medium], [Volume: «<1 ml»>]] ->`	"1 ml"
50-200 ul plasma was taken from heparinized blood.1 ml medium.	`TAKE: [[Reagent: heparinized blood.1 ml medium]] ->`	
Plasma was directly taken from cell culture.	`TAKE: [[Output: a plasma sample]] ->`	
Add 10 ul of the internal standard (10 uM C17-S1P in MeOH). Add 300 ul of 18.5 HCl.	`ADD: [[Reagent: 18.5 HCl], [Volume: «<300 ul»>]] ->`	"300 ul"
As an example, S1P extraction from a plasma sample is shown in step A7.	`SHOW: [[Output: step A7], [Reagent: a plasma sample]] ->`	
The CHCl3-phase is extracted by directly pipetting through the upper aqueous phase.	`EXTRACT: [[Output: the CHCl3], [Container: the upper aqueous phase], [Reagent: step A7]] ->`	
Add this CHCl3-phase to the transferred CHCl3-phase of step A7.	`ADD: [[Reagent: this CHCl3-phase]] ->`	
Vacuum-dry the CHCl3 in the vacuum rotator at 60 °C for 45 min.	`DRY: [[Reagent: «<the CHCl3»>], [Temperature: 60 °C], [Time: «<45 min»>]] ->`	"the CHCl3", "45 min"
Alternatively, the samples can be dried under nitrogen gas flow.	`DRY: [[Reagent: samples], [Time: 1-20 min]] ->`	
Re-equilibrate with 90 solution A.	`EQUILIBRATE: [[Output: S1P], [Concentration: 90 solution]] ->`	
S1P is analyzed with the mass transition 380 m/z -> 264 m/z. For quantitative analysis, a standard curve with S1P amounts of 1 pmol to 100 pmol as the internal standard is generated.	`EXAMINE: [[Output: quantitative analysis], [Reagent: S1P]] ->`	

Table A5: **Running cases on the semantics level regarding unknown unknowns**

Original Text	Semantic Level	Unknown Unknowns
Harvest approximately 1×10^7 cells by centrifugation for 5 min.	`HARVEST: [[Device: centrifugation], [Time: 5 min], [Force: «<2000 RPM»>]] ->`	"«<2000 RPM»>"
Cell lysates are homogenized by passing through 22-gauge needles.	`HOMOGENIZE: [[Reagent: cell lysates]] ->`	
Tubes are put on ice for 15 min to complete the lysis.	`INCUBATE: [[Container: tubes], [Time: 15 min], [Temperature: on ice]] ->`	
Crude extracts are then centrifuged.	`CENTRIFUGE: [[Force: «<2500 RPM»>], [Time: «<5 min»>]] ->`	"«<5 min»>"
Supernatants are transferred to fresh centrifuge tubes.	`TRANSFER: [[Container: fresh centrifuge tubes]] ->`	
Cold 5 M NaCl is added to each sample to make a salt concentration of between 0.7 – 1.0 M to disrupt protein-protein interactions.	`ADD: [[Container: each sample], [Reagent: 5 M NaCl], [Concentration: 0.7 - 1.0 M]] -> sample with NaCl`	
Spin the crude extracts by ultracentrifugation to properly pellet residual insoluble proteins from the extract.	`SPIN: [[Device: ultracentrifugation], [Force: «<55000 RPM»>], [Reagent: residual insoluble proteins]] -> Hypotonic Buffer`	"«<55000 RPM»>"
Transfer supernatants into fresh centrifuge tubes.	`TRANSFER: [[Reagent: supernatants], [Container: fresh centrifuge tubes]] ->`	
Rinse Protein A beads in Hypotonic Buffer until ready for use.	`RINSE: [[Reagent: Hypotonic Buffer]] -> use`	
Take a volume of cell lysates (prepared as described above).	`TAKE: [[Volume: cell lysates]] -> Hypotonic Buffer`	
Dilute with Hypotonic Buffer to 250 – 500 mM salt to enable protein-protein interactions.	`DILUTE: [[Reagent: Hypotonic Buffer]] -> antibody`	
Add 2 ug of preclearing antibody to the diluted lysate (e.g., anti), vortex, add 50 uL of Protein A beads.	`ADD: [[Reagent: antibody, Protein A beads]] -> polyclonal anti-MEKK1`	
Add 2 ug of polyclonal anti-MEKK1 to the lysates, add 50 uL of Protein A beads at 4 °C for 1 h.	`ADD: [[Reagent: polyclonal anti-MEKK1], [Container: the lysates], [Temperature: 4 °C], [Time: 1 h]] ->`	
Touchspin beads, wash beads with hypotonic buffer (supplemented with NaCl).	`WASH: [[Reagent: hypotonic buffer], [Concentration: «<300 mM»>]] ->`	"«<300 mM»>"
In total, 3 – 5 washes of the beads are performed.	`PERFORM: [[Reagent: hypotonic buffer], [Frequency: 3 - 5]] ->`	
Finally, wash once with Hypotonic Buffer.	`WASH: [[Reagent: Hypotonic Buffer]] ->`	
Purified MEKK1 may be stored by snap-freezing in liquid nitrogen.	`STORE: [[Method: snap-freezing], [Reagent: liquid nitrogen]] -> M`	
Following preparation of MEKK1 immunoprecipitates (as above), incubate with 7 ug of JNKK1(K131M) along with 5 uCi of [y-^{32}P]ATP for 30 min.	`INCUBATE: [[Reagent: JNKK1(K131M), [y-$^{32}$P]ATP], [Container: «<Kinase Assay Buffer»>], [Temperature: «<30 °C»>], [Time: 30 min]] ->`	"«<Kinase Assay Buffer»>", "«<30 °C»>"

Table A6: **Running cases on the execution level regarding capacity of resources**

Original Text	Execution Level	Reagent Flow Graph
Prepare annealing solution of 50 uM RNA/DNA oligos with 50 mM NaCl in DNase/RNase-free water, aliquot 50 ul in PCR tube.	`PREPARE: [[Output: annealing solution], [Concentration: 50 uM RNA/DNA oligos, 50 mM NaCl], [Reagent: DNase/RNase-free water], [Volume: 50 ul], [Container: PCR tube]] ->`	in: DNase/RNase-free water (50 ul), RNA/DNA oligos (50 uM), NaCl (50 mM); out: annealing solution (50 ul)
Dissolve inhibitor compound in DMSO to 10 mM, if needed, prepare serial dilutions in Milli-Q water.	`DISSOLVE: [[Output: inhibitor compound solution], [Reagent: inhibitor compound, DMSO]] ->`	in: inhibitor compound, DMSO; out: inhibitor compound solution (volume depends on dilution)
Add water (20 ul in blanks, 10 ul in controls) to 96-well plate.	`ADD: [[Output: water in wells], [Reagent: water], [Container: 96-well plate]] ->`	in: water (20 ul for blanks, 10 ul for controls); out: water in 96-well plate (20 ul in blanks, 10 ul in controls)
Add 80 ul RT reaction mix (1.25x).	`ADD: [[Output: RT reaction mix in wells], [Volume: 80 ul]] ->`	in: RT reaction mix (80 ul); out: RT reaction mix in 96-well plate (80 ul)
Add 10 ul inhibitor dilution to samples, to each well.	`ADD: [[Output: samples with inhibitor], [Volume: 10 ul], [Reagent: inhibitor dilution]] ->`	in: inhibitor dilution (10 ul); out: samples with inhibitor (10 ul)
Stop reaction with 50 ul EDTA (0.5 M, pH 8.0).	`STOP: [[Output: stopped reaction], [Reagent: EDTA], [Volume: 50 ul]] ->`	in: EDTA (50 ul); out: stopped reaction with EDTA (50 ul)
Quantify reaction with Victor 3 at 490/528 nm, report inhibitor values as percentage of control.	`QUANTIFY: [[Output: quantified reaction], [Device: Victor 3]] ->`	in: reaction; out: quantified reaction at 490/528 nm
Subtract blank value from samples.	`SUBTRACT: [[Output: corrected samples], [Reagent: blank value]] ->`	in: blank value, samples; out: corrected sample values
Calculate IC50 value as the concentration reporting 50 reduction of signal compared to control.	`CALCULATE: [[Output: IC50 value], [Reagent: signal]] ->`	in: signal; out: IC50 value

Table A7: **Running cases on the execution level regarding safety of operations**

Original Text	Execution Level	Reagent Flow Graph
Replace medium after 12 hours (Day 2).	`REPLACE: [[Output: medium replaced], [Container: medium]] ->`	in: old medium; out: new medium
Digest mESCs with 0.05 trypsin, prepare for FACS into 96-well plates (Day 10).	`DIGEST: [[Output: mESCs], [Reagent: 0.05 trypsin], [Container: 96-well plates]] ->`	in: mESCs, 0.05 trypsin; out: digested mESCs (ensure trypsin is neutralized to avoid over-digestion)
Remove single colonies from 96-well plates to 24-well plates.	`REMOVE: [[Output: single colonies], [Container: 96-well plates, 24-well plates]] ->`	in: single colonies; out: single colonies in 24-well plates
Confirm positive colonies by transient transfection of sgRNAs analysis (SPH primers) (Day 14-15).	`CONFIRM: [[Output: positive colonies], [Reagent: SPH primers]] ->`	in: single colonies, SPH primers; out: positive colonies
Sort single cells into 96-well plates by FACS.	`SORT: [[Output: single cells], [Device: FACS], [Container: 96-well plates]] ->`	in: single cells; out: sorted single cells in 96-well plates (ensure proper calibration of FACS to avoid sorting errors)
Confirm insertion by PCR (Day 18).	`CONFIRM: [[Output: insertion confirmed]] ->`	in: single cells; out: confirmed insertion
Confirm positive colonies by PCR (Day 22).	`CONFIRM: [[Output: positive colonies]] ->`	in: single colonies; out: positive colonies
Measure fluorescent intensity of colonies by FACS, take fluorescence images under confocal microscope (Day 27).	`TAKE: [[Output: fluorescence images], [Device: confocal microscope], [Container: colonies]] ->`	in: colonies; out: fluorescence images (handle samples to avoid photobleaching)

E.3 Types of errors made by our translator

We present a detailed analysis of the errors made by our proposed automatic translator compared to human experts. We discuss the potential improvements of the translator accordingly.

Distinctions on the syntax level Difference between the translation results of our system and those of experts is subtle, with the biggest difference being in the analysis of long sentences in natural language. For human experts, it is natural and easy to analyze the parameters of events/actions or multiple actions in long sentences, while for our approach, there are sometimes problems with the correspondence between action and parameter, which need to be improved in future work.

This series of examples in Tab. A8 demonstrates the superior performance of our system at the syntax level when processing relatively short sentences.

Table A8: **Distinctions on the syntax level when processing relatively short sentences**

Original Text	Syntax Level	Remarks
Transfer 200 ul of serum into a 1.5 ml microfuge tube.	TRANSFER: [[Output: buffer], [Reagent: serum], [Container: a 1.5 ml microfuge tube], [Volume: 200 ul]] ->	
Add 200 ul of buffer XBP.	ADD: [[Output: the sample/XBP mixture], [Reagent: buffer], [Volume: 200 ul]] ->	
Add the sample/XBP mixture onto the exoEasy spin column.	ADD: [[Output:], [Reagent: the sample/XBP mixture]] ->	
Add 10 ml of buffer XWP to the exoEasy spin column.	ADD: [[Output:], [Reagent: buffer XWP], [Volume: 10 ml]] ->	
Transfer the spin column to a fresh collection tube.	TRANSFER: [[Output: Qiazol], [Container: a fresh collection tube]] ->	
Add 700 ul Qiazol to the membrane of the spin column.	ADD: [[Output:], [Reagent: Qiazol], [Volume: 700 ul]] ->	
Transfer the flow-through, which is the lysate, to a PLG tube.	TRANSFER: [[Output: flow-through], [Container: PLG tube]] ->	
Incubate at room temperature for 5 minutes.	INCUBATE: [[Output:], [Temperature: room temperature], [Time: 5 minutes]] ->	
Add 90 ul chloroform.	ADD: [[Output:], [Volume: 90 ul]] ->	

This series of examples in Tab. A9 illustrates the challenges faced with longer sentences due to the diversity of actions and the multiple parameters.

Table A9: **Distinctions on the syntax level when processing longer sentences**

Original Text	Syntax Level	Remarks
Rinse MSC with 2 ml PBS.	RINSE: [[Output: MSC with 2 ml PBS], [Reagent: PBS], [Volume: 2 ml]] ->	
Trypsinize with 0.5 ml trypsin.	TRYPSINIZE: [[Output:], [Reagent: trypsin], [Volume: 0.5 ml]] ->	
Transfer to 15 ml tube.	TRANSFER: [[Output:], [Container: 15 ml tube]] ->	
Add 10 ml DMEM with 10 FBS.	ADD: [[Output:], [Reagent: DMEM, FBS], [Volume: 10 ml]] ->	
Incubate overnight at 37 °C, 5 CO2.	INCUBATE: [[Output:], [Temperature: 37 °C]] ->	Lack of parameter
Wash cells twice with 2 ml PBS, add osteogenic differentiation medium.	WASH: [[Output:], [Volume: 2 ml PBS], [Reagent: osteogenic differentiation medium]] ->	Lack of action in single sentence
Change medium every 2 days for 10 days.	CHANGE: [[Output: Alizarin red S], [Time: every 2 days for 10 days]] ->	
At day 10, stain with Alizarin red S for 5 min.	STAIN: [[Output:], [Reagent: Alizarin red S], [Time: 5 min]] ->	

Distinctions on the semantics level When supplementing known unknowns, human experts tend to rely on contextual reasoning. Since experts are not familiar with protocols from all fields, they often infer parameters based on context for protocols outside their expertise. The primary source of their errors is a lack of understanding of protocols in specific domains, which is fundamentally consistent with the approach of our system. When supplementing unknown unknowns, human experts tend to transfer their knowledge from familiar domains, such as instruments used or common parameters, to protocols in various fields, using this as a basis for parameter supplementation. Our system, however, completes parameters based on all collected protocols, which is essentially the opposite of the transfer process used by human experts.

The example presents as follows — the completion of two types of parameters at the semantic level is included: for instance, determining the configuration parameter for an operation, where human experts rely on personal experimental experience; and inferring the required reagents for one step, where human experts use contextual reasoning. When the context is not sufficiently clear, human experts cannot infer the known unknowns within a single sentence.

Table A10: **Machine results on the semantics level**

Original Text	Machine Result	Remarks
Add 700 ul of buffer RWT to the RNeasy MinElute spin column.	ADD: [[Output:], [Reagent: «<buffer RWT»>], [Volume: 700 ul]] ->	known unknown
Discard the flow-through.	DISCARD: [[Output: the flow-through], [Volume:]] ->	
Discard the collection tube with the flow-through.	DISCARD: [[Output: the flow-through], [Container: the collection tube], [Volume:], [Reagent: the flow-through]] ->	
Transfer the RNeasy MinElute spin column into a new 2 ml collection tube (supplied).	TRANSFER: [[Output:], [Device: RNeasy MinElute], [Container: a new 2 ml collection tube (supplied)], [Volume:]] ->	
Open the lid of the spin column.	OPEN: [[Output:], [Container: «<spin column»>]] ->	known unknown
Centrifuge at full speed (14,000 xg) to dry the membrane.	CENTRIFUGE: [[Output:], [Speed: full speed (14,000 xg)], [Container: membrane], [Time: «<5 min»>]] ->	unknown unknown
Discard the collection tube with the flow-through.	DISCARD: [[Output: the flow-through], [Container: the collection tube], [Volume:], [Reagent: the flow-through]] ->	
Transfer the RNeasy MinElute spin column into a new 1.5 ml collection tube.	TRANSFER: [[Output: RNase-free water], [Device: RNeasy MinElute], [Container: a new 1.5 ml collection tube (supplied)], [Volume:]] ->	
Add 14 ul RNase-free water directly to the center of the spin column membrane.	ADD: [[Output:], [Reagent: «<RNase-free water»>], [Volume: 14 ul]] ->	known unknown

Table A11: **Human results on the semantics level**

Original Text	Human Study Result	Remarks
Add 700 ul of buffer RWT to the RNeasy MinElute spin column.	ADD: [[Output:], [Reagent: «<NONE»>], [Volume: 700 ul]] ->	known unknown
Discard the flow-through.	DISCARD: [[Output: the flow-through], [Volume:]] ->	
Discard the collection tube with the flow-through.	DISCARD: [[Output: the flow-through], [Container: the collection tube], [Volume:], [Reagent: the flow-through]] ->	
Transfer the RNeasy MinElute spin column into a new 2 ml collection tube (supplied).	TRANSFER: [[Output:], [Device: RNeasy MinElute], [Container: a new 2 ml collection tube (supplied)], [Volume:]] ->	
Open the lid of the spin column.	OPEN: [[Output:], [Container: «<spin column»>]] ->	known unknown
Centrifuge at full speed (14,000 xg) to dry the membrane.	CENTRIFUGE: [[Output:], [Speed: full speed (14,000 xg)], [Container: membrane], [Time: «<5 min»>]] ->	unknown unknown
Discard the collection tube with the flow-through.	DISCARD: [[Output: the flow-through], [Container: the collection tube], [Volume:], [Reagent: the flow-through]] ->	
Transfer the RNeasy MinElute spin column into a new 1.5 ml collection tube.	TRANSFER: [[Output: RNase-free water], [Device: RNeasy MinElute], [Container: a new 1.5 ml collection tube (supplied)], [Volume:]] ->	
Add 14 ul RNase-free water directly to the center of the spin column membrane.	ADD: [[Output:], [Reagent: «<water»>], [Volume: 14 ul]] ->	known unknown

Distinctions on the execution level Human experts track capacity primarily based on prior knowledge, subsequently using context to judge the appropriateness of the equipment used. In contrast, the machine extracts the entire flow process, enabling it to calculate each step and ensure that the capacity tracking is scientifically sound and reasonable.

This series of examples in Tab. A12 demonstrates how our system tracks the required capacities at each step of the protocol by contextualizing the step into the spatial dimension.

Table A12: **Distinctions on the execution level (spatial dimension)**

Original Text	Execution Level	Resources
Add 4 ul of 160 mM KMnO4 to radiolabeled DNA (40 ng, 5,000-10,000 cpm) in 40 ul total volume.	ADD: [[Output: reaction mixture], [Reagent: 160 mM KMnO4, radiolabeled DNA], [Volume: 4 ul, 40 ul]] ->	"radiolabeled DNA"
Precipitate with ethanol.	PRECIPITATE: [[Output: precipitate], [Reagent: ethanol]] ->	"reaction mixture"
Dissolve in 70 ul 10 piperidine.	DISSOLVE: [[Output: dissolved DNA], [Reagent: 10 piperidine], [Volume: 70 ul]] ->	"precipitate"
Incubate at 90 °C for 30 min.	INCUBATE: [[Output: incubated DNA], [Temperature: 90 °C], [Time: 30 min]] ->	"dissolved DNA"
Precipitate with ethanol.	PRECIPITATE: [[Output: pellets], [Reagent: ethanol]] ->	"incubated DNA"
Wash pellets with 70 ethanol, dry, dissolve in 5 ul electrophoresis loading buffer.	RINSE: [[Output: non-labeled DNA], [Reagent: 70 ethanol, electrophoresis loading buffer], [Volume: 5 ul]] ->	"pellets"

This series of examples in Tab. A13 illustrates how our system tracks the preconditions and post-conditions at each step of the protocol by contextualizing the step into the temporal dimension.

Table A13: **Distinctions on the execution level (temporal dimension)**

Original Text	Execution Level	Resources
Freeze cells for 1 hour at -80°C, thaw at 37°C for 1 hour.	FREEZE: [[Output: DLFR004], [Reagent: cells], [Time: 1 hour], [Temperature: -80°C]], THAW: [[Output: DLFR004], [Reagent: cells], [Time: 1 hour], [Temperature: 37°C]] ->	"DLFR004"
If not using DLFR004, lyse cells with lysis buffer.	LYSE: [[Output: cell lysate], [Reagent: lysis buffer], [Condition: not using DLFR004]] ->	"DLFR004"
Prepare serological pipette by cutting at the 3 mL mark, sealing bottom with parafilm.	PREPARE: [[Output: modified pipette], [Device: serological pipette], [Modification: cutting at the 3 mL mark, sealing bottom with parafilm]] ->	"lysis buffer"
Secure serological pipette to a vertical surface.	SECURE: [[Output: secured pipette], [Device: serological pipette]] ->	"modified pipette"
Fill pipette with at least 2.5 mL cell lysate, measure distance from 2 mL to 1 mL mark.	FILL: [[Output: filled pipette], [Volume: at least 2.5 mL], [Reagent: cell lysate]] ->	"secured pipette"
Position cell phone camera to record pipette, drop a glass bead inside, repeat two more times.	POSITION: [[Output: recorded experiment], [Device: cell phone camera, pipette], [Reagent: glass bead]] ->	"filled pipette"
Remove parafilm seal.	REMOVE: [[Output:], [Container: parafilm seal]] ->	"recorded experiment"
Rinse pipette.	RINSE: [[Output: cleaned pipette], [Device: pipette]] ->	"next sample"
Repeat with next sample to obtain triplicates.	REPEAT: [[Output: triplicates]] ->	"triplicates"

E.4 Properties of the pre-processing pipeline

Significant differences exist between various stages of the pre-processing pipeline. We present several real-world examples to illustrate these distinctions in Tab. A14.

Table A14: **Showcases of action extraction, entity extraction, and classification in protocol steps**

Original Text	Action Extraction	Entity Extraction	Classification with LLM	Preprocess Result	LLM-Pure
Stain with DAPI nucleic acid stain for 30 seconds.	stain	[Reagent: DAPI nucleic acid stain], [Time: 30 seconds]	[Reagent: DAPI nucleic acid stain], [Time: 30 seconds]	STAIN: [[Output:], [Reagent: DAPI nucleic acid stain], [Time: 30 seconds]]	STAIN: [[Duration: 30 seconds], [Reagent: DAPI nucleic acid stain]]
Purify CD4+ by magnetic isolation using the Auto MACS sorter (Miltenyi Biotec) using POSSELD2 program.	purify	[Reagent: CD4+], [Device: the Auto MACS sorter (Miltenyi Biotec)], [Device: POSSELD2 program]	[Reagent: CD4+], [Device: the Auto MACS sorter (Miltenyi Biotec)], [Device: POSSELD2 program]	PURIFY: [[Device: the Auto MACS sorter (Miltenyi Biotec), POSSELD2 program], [Output:], [Reagent: CD4+]]	PURIFY: [[Device: the Auto MACS sorter (Miltenyi Biotec), POSSELD2], [Method: magnetic isolation], [Reagent: CD4+]]
Measure baseline oxidative status every 20 s for at least 5 min, then add stimulating substances (e.g., thapsigargin).	measure, add	[Output: baseline oxidative status], [Time: every 20 s], [Time: 5 min], [Reagent: stimulating substances]	[Output: baseline oxidative status], [Time: every 20 s], [Time: 5 min], [Reagent: stimulating substances]	MEASURE: [[Output: baseline oxidative status], [Time: every 20 s, 5 min]]; ADD: [[Reagent: stimulating substances]]	MEASURE: [[Output: baseline oxidative status], [Reagent: stimulating substances], [Time: every 20 s, 5 min]]
Spin the crude extracts by ultra-centrifugation at 55000 RPM to properly pellet residual insoluble proteins from the extract.	spin	[Device: ultracentrif [Force: 55000 RPM], [Reagent: residual insoluble proteins], [Container: the extract]	[Device: ultracentrifugatio [Force: 55000 RPM], [Reagent: residual insoluble proteins], [Container: the extract]	SPIN: [[Device: ultra-centrifugation], [Force: 55000 RPM], [Output:], [Reagent: residual insoluble proteins]]	SPIN: [[Reagent: crude extracts], [Method: ultra-centrifugation], [Purpose: to properly pellet residual insoluble proteins from the extract], [Speed: 55000 RPM]]
Confirm positive colonies by transient transfection of sgRNAs analysis (SPH primers).	confirm	[Output: positive colonies], [Reagent: sgRNAs analysis (SPH primers)]	[Output: positive colonies], [Reagent: sgRNAs analysis (SPH primers)]	CONFIRM: [[Output: positive colonies], [Reagent: sgRNAs analysis (SPH primers)]]	CONFIRM: [[Device: SPH primers], [Method: transient transfection of sgRNAs analysis], [Output: positive colonies]]

F Reproducibility

The project page with supplementary files for reproducing the results of this paper will be available at `https://autodsl.org/procedure/papers/neurips24shi.html`.

G Limitations

As a systematic study with a proof-of-concept framework, the design and evaluation of the pipeline come with limitations, leading to further investigations:

- We majorly exploit the approaches of empirical study to observe the behavior of DSLs and human experts for extracting design principles. Can we draw theories from information theory to rigorously prove the expression capacity of DSLs and other structural knowledge representations, to advance our design choice?
- We majorly consider the imperative programming DSLs as the vehicle of PDGs in this work. This raises the question of whether incorporating alternative programming paradigms, such as functional and object-oriented models, could enhance the representation of complex entities within protocols, particularly the properties of reagents.
- Can we extend the protocol translator to a larger set of experiments, especially those with heterogeneous hardware devices such as mobile robots?
- Can we find similar mechanism in other critical domains with the requirements on protocol execution, such as advanced manufacturing, and generalize our translator for such applications?

With many questions unanswered, we hope to explore more on automated protocol translation for self-driving laboratories and beyond.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In this paper, we systematically study the problem of translating protocols for human experimenters into those suitable for self-driving laboratories, in order to standardize and automate the translation process. Further, we propose the initial proof-of-concept framework that fully frees domain experts from hand-crafting protocol translators.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We have discussed the potential limitations at Appx. G.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: No theoretical result is included in this paper.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided them with the implementation details at Appx. C. We will also release our codes upon acceptance.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We have provided them with the implementation details Appx. C. We will also release our codes upon acceptance.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: All of them are carefully illustrated in implementation details Appx. C.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Yes. Please refer to Fig. 3.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Please refer to Appx. C.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Yes. Please refer to the main text.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Please refer to the general discussions at Sec. 5.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes. Their licenses are checked and their published works are properly cited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: Please refer to Appx. B and Appx. D.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [Yes]

    Justification: We have obtained an approved IRB in advance. Please refer to Appx. B.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.