# Process-Level Representation of Scientific Protocols with a Text-Based Game Annotation Interface

**Ronen Tamari***
The Hebrew University of Jerusalem
ronent@cs.huji.ac.il

**Fan Bai**
Georgia Institute of Technology
fan.bai@cc.gatech.edu

**Alan Ritter**
Georgia Institute of Technology
alan.ritter@cc.gatech.edu

**Gabriel Stanovsky***
The Hebrew University of Jerusalem
gabis@cs.huji.ac.il

## Abstract

We develop Process Execution Graphs (PEG), an executable document-level representation of real-world wet lab biochemistry protocols, addressing challenges such as cross-sentence relations, long-range coreference, grounding, and implicit arguments. We built a corpus of complex lab protocols with a novel interactive simulator built upon a text-based game engine that keeps track of entity traits and semantic constraints during annotation, yielding high quality annotated PEGs. Our framework presents several directions for future work, including the modelling of challenging long range dependencies, application of text-based games for real-world procedural text understanding, and extending simulation-based annotation to new domains. [2]

## 1 Introduction

There is a drive in recent years towards automatic wet lab environments, where menial benchwork procedures such as pipetting, centrifuging, or incubation are carried out automatically by software-controlled lab equipment. These environments allow reliable and precise experiment reproducbility while relieving researchers from tedious and laborious work which is prone to human error [Bates et al., 2017, Keller et al., 2019]. To achieve this, several programmatic formalisms are developed to describe an experiment as an executable program. For example, Autoprotocol [Lee and Miles, 2018] defines a `mix` predicate taking three arguments: *mode*, *speed*, and *duration*.

A promising direction to leverage automatic wet-lab environments is a conversion from natural language protocols, written in expressive free-form language, to low-level, machine executable instructions, ensuring a non-ambiguous, repeatable description of experiments [Mehr et al., 2020].

However, a large semantic gap exists between machine-executable formalisms and standard annotation methods for scientific procedural text (§4). Current datasets, such as Wet Labs Protocols (WLP) [Kulkarni et al., 2018] and Materials Science Procedural Text Corpus (MSPTC) [Mysore et al., 2019], annotate *action-graph* semantics; sentence-level, shallow semantic structures consisting of labeled text-spans and relations between them (Fig. 1, top right). While these annotations are useful for text-mining, they do not provide sufficient process-level details for execution; for

*Work begun at the Allen Institute for Artificial Intelligence.

[2]We will make our annotated corpus, simulator, annotation interface and interaction data available for use by the research community.

Wordplay: When Language Meets Games Workshop @ NeurIPS 2020,
https://wordplay-workshop.github.io/

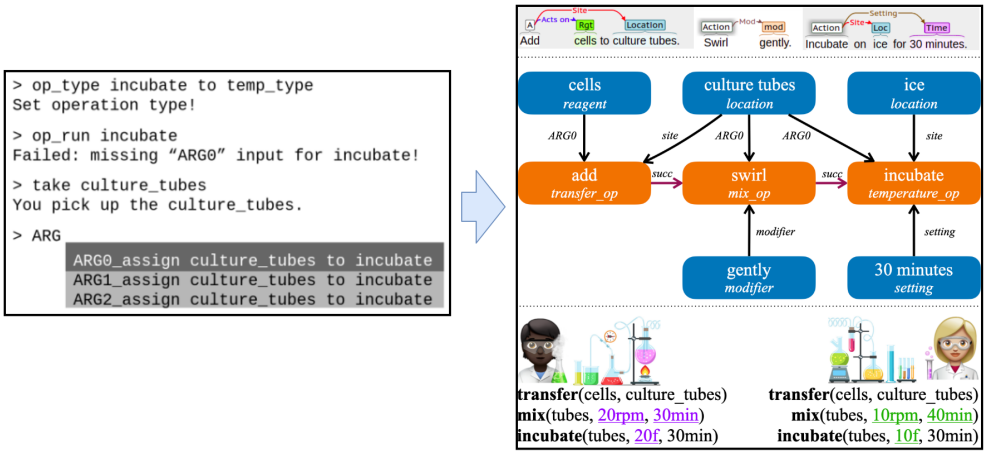


Figure 1: We use text-based games as an interactive annotation interface (left) to capture Process Execution Graphs (right, middle): grounded, process level representations bridging sentence-level action-graph representations (right,top) and lab-specific executable instructions (right, bottom). Our interactive framework and data enable development of text-based game agents as text-to-code virtual assistants.

example, resolution of implicit arguments and structured representation of entity locations throughout a procedure [Mehr et al., 2020].

In this work, we focus on a crucial step bridging natural language protocols and executable instructions – the extraction and representation of the relations conveyed by the protocol in a formal graph structure, termed Process Execution Graphs (PEG), exemplified in Figure 1. PEGs capture both concrete, exact quantities ("*30 minutes*"), as well as vague instructions ("swirl *gently*"). A researcher can then port the PEG (either manually or automatically) to their specific lab equipment, e.g., specifying what constitutes a gentle swirl setting and adding missing arguments, such as the duration of the incubation in Figure 1.

Formally, PEGs are directed, acyclic labeled graphs (§2), capturing how objects in the lab (e.g., *cells*, *tubes*) are manipulated by lab operations (e.g., `mixing`, `incubating`), and in what order. Importantly, PEGs capture relations which may span across multiple sentences and implicit arguments. For example, the PEG in Figure 1 explicitly captures the relation between *culture tubes*, mentioned in the first sentence, and `swirl` and `incubate` which appear in later sentences.

To annotate long and complex lab protocols, we develop a text-based game annotation interface (Fig. 1) simulating objects and actions in a lab environment (§3). Our annotators are given natural language wet labs procedures from WLP, and are asked to repeat their steps by issuing textual commands to the simulator. The commands are deterministically converted to our PEG representation. This interface takes much of the burden off annotators by keeping track of object traits and commonsense constraints. For example, when the annotator issues a single `transfer` command for a container, the simulator moves all its contents as well. We find that in-house annotators were able to effectively use this interface on complex protocols, achieving good agreement and capturing rich detail compared with span-based annotation methods (e.g., BRAT [Stenetorp et al., 2012]).

In conclusion, we make the following contributions: (1) We formalize a PEG representation for free-form, natural language lab protocols, providing a semantic scaffold between free-form scientific literature and low-level, machine executable instructions. (2) We develop a novel annotation interface for procedural text annotation using text-based games, and show that it is intuitive enough for wet-lab protocol annotation. (3) We release a challenging corpus of 279 PEGs representing document-level lab protocols from WLP. This size is on par with similar corpora of procedural text [Mysore et al., 2019, Vaucher et al., 2020], while providing additional process-level information hitherto unavailable.

## 2 Task Definition: Process Execution Graphs

Intuitively, we extend the WLP annotations [Kulkarni et al., 2018] from the sentence level to entire documents, aiming to capture *all* of the relations in the protocol. Formally, our representation is a directed, labeled, acyclic graph structure, dubbed a Process Execution Graph (PEG), exemplified in Figures 1 and 2, and formally defined below. See Appendix A for the complete ontology.

### 2.1 PEG Structure

**Nodes.** PEG nodes are triggered by explicit text spans in the protocol, appearing in the center of each node (e.g., "swirl", or "ice"). Nodes consist of two types: (1) *predicates*, marked in orange: denoting lab operations, such as `add`; and (2) *arguments*, marked in blue: representing physical lab objects (e.g., *culture tubes*, *cells*), exact quantities (*30 minutes*), or abstract instructions (e.g., *gently*).

**Edges.** Following PropBank notation [Kingsbury and Palmer, 2003], PEGs consist of three types of edges derived from the Autoprotocol ontology, and denoted by their labels: (1) core-roles (e.g., "ARG0", "ARG1"): indicating predicate-specific roles, aligning with Autoprotocol's ontology. For example, *ARG0* of `mix` assigns the element to be mixed; (2) non-core roles (e.g., "setting", "site", or "co-ref"): indicate predicate-agnostic relations. For example, the *site* argument always marks the location in which a predicate is taking place; and (3) temporal edges, labeled with a special "succ" label: define a temporal transitive ordering between predicates. In Figure 1, `add` occurs before `swirl`, which occurs before `incubate`.

### 2.2 Comparison with action graphs

Aside from their inherent support for execution, PEGs differ from action graph annotations in three main aspects.

**Operation types.** To support action-specifc semantics, PEGs record fine-grained operation types based on Autoprotocol (e.g., `transfer`, `mix`), beyond the generic `action` span label used in WLP and MSPTC.

**Argument re-use.** While the PEG does not form directed cycles (since temporal relations define a partial ordering), it does form non-directed cycles (or *re-entrancies*). These occur where there exist nodes $u, v$ such that there are two different paths from $u$ to $v$. This occurs when an object is re-used, i.e., participates in two or more temporally-dependent operations. For example, see *culture tubes*, which participates in all operations in Figure 1. Action-graph approaches typically do not annotate argument re-entrancies owing to complications in the case of operations that cause a materials' state to change. As noted in Mysore et al. [2019], "when a materials state changes due to a specific operation, considering the same text span to be the argument of a different operation would not be chemically valid." Our simulator-based approach allows us to address this difficulty by abstracting away from text spans: spans function as references to objects for which state-change can be tracked. Furthermore, as can be seen for the case of the *tubes* in Fig. 1, re-entrancies feature centrally in typical machine-executable protocol representations, in the form of calls by reference. Therefore, re-entrancies should be accounted for to capture the full process workflow necessary in an executable setting.

**Cross-sentence relations.** Edges $(u, v)$ may be triggered either by *within-sentence relations*, when both $u$ and $v$ are triggered by spans in the same sentence, or by *cross-sentence relations*, when $u$ and $v$ are triggered by spans in different sentences. Previous efforts have focused on sentence-level annotation for simplicity [Vaucher et al., 2020], and excluded synthesis procedures featuring primarily cross-sentence relations [Mysore et al., 2019]. While cross sentence relations typically complicate span-based methods, we hypothesize that our simulator makes annotation of such relations more intuitive; unlike span-based representation, simulators can more naturally represent concrete referents which persist across sentences unless mentioned otherwise.

In the following section, we will show that both reentrancies and cross-sentence relations are abundant and annotated with good agreement in our dataset.

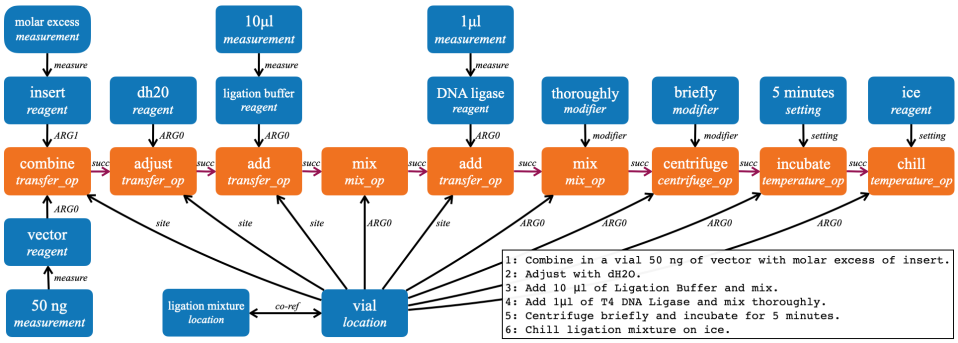


Figure 2: A gold PEG annotation for a real-world wet lab protocol whose text is presented in the lower right corner, exemplifying several common properties: (1) complex language, in relatively short sentences; (2) a chain of temporally-dependent, cross-sentence operations; (3) a common object that is being acted upon through side effects throughout the process (*vial*); and (4) *vial* is mostly omitted in the text after being introduced in the first sentence, despite participating in all following sentences. In the last sentence it appears with a metonymic expression (*ligation mixture*).

## 3 Data Collection: The X-WLP Corpus

In this section, we describe in detail the creation of our annotated corpus: X-WLP. The protocols in X-WLP are a subset (44.8%) of those annotated in the WLP corpus. These were chosen because they are covered well by Autoprotocol's ontology.

In total, we collected 4,262 sentences (54.1K tokens) in 279 wet lab protocols annotated with our graph representation. X-WLP annotates long examples, often spanning dozens of sentences, and its size is comparable (e.g., in terms of annotated words) to the MSPTC corpus [Mysore et al., 2019] and other related procedural datasets. See Appendix B.1 for a detailed comparison.

Despite WLP's focus on sentence-level relations (see top of Figure 1), it is a valuable starting point for a document-level representation. We pre-populate our PEG representations with WLP's gold object mentions (e.g., *cells*, *30 minutes*), operation mentions (`swirl` and `incubate`), and within-sentence relations (e.g., between *gently* and `swirl`). We then ask annotators to enrich them with type grounding for operations and arguments, as well as cross-sentence relations, as defined in §2. From these annotations we obtain process-level representations such as those presented in Figures 1 and 2.

### 3.1 Process-Level Annotation Interface: Text-Based Simulator

Annotating cross-sentence relations and grounding without a dedicated user interface is an arduous and error-prone prospect. Consider as an example the *ligation mixture* mention in Figure 2. This mention is a metonym for *vial* (5 sentences earlier), after mixing in the *ligase*. This kind of metonymic co-reference is known to be difficult for annotation [Jurafsky and Martin, 2009], and is cumbersome to represent using span-based methods like BRAT [Stenetorp et al., 2012]. A simulator can provide a natural way to account for it by representing the relevant temporal and contextual information: after sentence 4, *vial* contains the *ligation buffer* mixed with other entities.

To overcome these challenges and achieve high-quality annotations for this complex task, we develop a simulator annotation interface, building upon the TextWorld framework [Côté et al., 2018]. This approach uses text-based games as the underlying simulator environment, which we adapt to the biochemistry domain.

The annotator interacts with the text-based interface to simulate the raw wet lab protocol (Figure 1): setting the types of operations (the first interaction sets the span "incubate" as a `temperature` operation) and assigning their inputs (the last line assigns `culture tubes` as an input to `incubate`), while the simulator tracks entity states and ensures the correct number and type of arguments, based on the Autoprotocol ontology. For example, the second interaction in Figure 1 indicates a missing argument for the `incubate` operation (the argument to be incubated). Finally,

tracking temporal dependency ("succ" edges) is also managed entirely by the simulator by tracking the order in which the annotator issues the different operations.

Further assistance is provided to annotators in the form of of an auto-complete tool (last interaction in Figure 1), visualization of current PEG and a simple heuristic "linter" [Johnson, 1977] which flags errors such as ignored entities, by producing a score based on the number of connected components in the output PEG.

### 3.2 Data Analysis

**Inter-annotator agreement.** We turn to the literature on abstract meaning representation (AMR; Banarescu et al., 2013) for established graph agreement metrics using the Smatch score [Cai and Knight, 2013], which we adapt to our setting. We report a mean Smatch of $0.84$, comparable to those obtained for AMR, where reported gold agreement varies between $0.69 - 0.89$ [Cai and Knight, 2013]. See Appendix B.2 for detailed analysis.

**Information gain from process-level annotation.** Analysis of the relations in X-WLP reveals that a significant proportion of arguments in PEGs are re-entrancies (32.4%) or cross-sentence (50.3%)[3]. Figure 2 shows a representative example, with the *vial* participating in multiple re-entrancies and long-range relations, triggered by each sentence in the protocol.

To shed light on the additional process-level information captured by X-WLP relative to WLP, we compare (X-WLP, WLP) the average number of arguments per operation node ($3.01$, $1.87$) as well as the ratio of operation nodes with no core arguments ($0.0$, $0.19$). For example, see the `swirl` instruction at the top of Figure 1: in WLP, this predicate has no core role argument and is thus semantically under-defined. X-WLP correctly captures the core role of *culture tubes*. By definition, our use of input validation by the simulator prevents semantic under-specification, which is likely a significant factor in the higher counts for cross-sentence relations and overall average arguments in X-WLP.

## 4 Related & Future Work

As discussed above, our work builds upon sentence-level shallow semantic parsing approaches to annotating procedural text, including WLP [Kulkarni et al., 2018], biology textbooks [Berant et al., 2014], materials science [Mysore et al., 2019]. These approaches have focused mainly on text mining applications, whereas outputting a machine executable synthesis procedure requires a more structured, process-level semantic representation.

Recent works are beginning to address this requirement: Vaucher et al. [2020] converts chemical synthesis procedures to more structured action representations for downstream incorporation into automated workflows. However, annotation is still only at sentence level. Concurrently with this work, Mehr et al. [2020] have similarly proposed a process-level executable representation for chemical procedures. They also focus primarily on sentence level analysis and rely on a human-in-the-loop to verify the process-level conversion. Future work could leverage our approach towards enhancing process-level parsers.

Structurally, PEGs are similar to abstract meaning representation (AMR; Banarescu et al. 2013), allowing us to use agreement and performance metrics developed for AMR. In contrast with the sentence-level AMR, a major challenge in this work is annotating procedure-level representations.

Existing process-level datasets have been limited to shorter and simpler texts; ProPara [Dalvi et al., 2018] contains short (non-expert) annotations of scientific processes. Kiddon et al. [2015] and Bosselut et al. [2018] provide a small number of gold process-level annotations for cooking recipes.

Our framework also provides a link to text-based game approaches to procedural text understanding. Tamari et al. [2019] modelled scientific procedures with text-based games but used only synthetic data. Our simulator enables leveraging recent advances on text-based games agents (e.g., [Adhikari et al., 2020]); applying such agents towards complex natural language understanding is a promising future direction.

[3] See Appendix B.3 for detailed analysis.

## Broader Impact

Our work focuses on enhancing the capability for machine reading of scientific experimental protocols. This research may benefit a wide range of scientific communities where much research remains "trapped" in unstructured textual format, which due to its ambiguous nature, presents many challenges for reproduceability and knowledge transfer. More precise machine reading technology may also be used maliciously, for example to harvest structured information for surveillance purposes.

## References


Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and William L. Hamilton. Learning Dynamic Knowledge Graphs to Generalize on Text-Based Games. 2020. URL http://arxiv.org/abs/2002.09127.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. Abstract meaning representation for sembanking. In *LAW@ACL*, 2013.

Maxwell Bates, Aaron J Berliner, Joe Lachoff, Paul R Jaschke, and Eli S Groban. Wet lab accelerator: a web-based application democratizing laboratory automation for synthetic biology. *ACS synthetic biology*, 6(1):167–171, 2017.

Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. Modeling biological processes for reading comprehension. In *EMNLP*, 2014.

Antoine Bosselut, Corin Ennis, Omer Levy, Ari Holtzman, Dieter Fox, and Yejin Choi. Simulating action dynamics with neural process networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJYFzMZC-.

Shu Cai and Kevin Knight. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P13-2131.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, et al. Textworld: A learning environment for text-based games. In *Workshop on Computer Games*, pages 41–75. Springer, 2018.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. Tracking state changes in procedural text: a challenge dataset and models for process paragraph comprehension. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1595–1604, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1144. URL https://www.aclweb.org/anthology/N18-1144.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. An incremental parser for Abstract Meaning Representation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 536–546, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-1051.

Stephen C Johnson. *Lint, a C program checker*. Citeseer, 1977.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Pearson Prentice Hall, second edition, 2009.

Ben Keller, Justin Vrana, Abraham Miller, Garrett Newman, and Eric Klavins. Aquarium: The Laboratory Operating System version 2.6.0, March 2019. URL https://doi.org/10.5281/zenodo.2583232.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. Mise en place: Unsupervised interpretation of instructional recipes. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 982–992, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1114. URL https://www.aclweb.org/anthology/D15-1114.

Paul Kingsbury and Martha Palmer. Propbank: the next level of treebank. 2003.

Chaitanya Kulkarni, Wei Xu, Alan Ritter, and Raghu Machiraju. An annotated corpus for machine reading of instructions in wet lab protocols. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 97–106, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2016. URL https://www.aclweb.org/anthology/N18-2016.

Peter L Lee and Benjamin N Miles. Autoprotocol driven robotic cloud lab enables systematic machine learning approaches to designing, optimizing, and discovering novel biological synthesis pathways. In *SIMB Annual Meeting 2018*. SIMB, 2018.

S. Hessam M. Mehr, Matthew Craven, Artem I. Leonov, Graham Keenan, and Leroy Cronin. A universal system for digitization and automatic execution of the chemical synthesis literature. *Science*, 370(6512):101–108, 2020. ISSN 0036-8075. doi: 10.1126/science.abc2986. URL https://science.sciencemag.org/content/370/6512/101.

Sheshera Mysore, Zachary Jensen, Edward Kim, Kevin Huang, Haw-Shiuan Chang, Emma Strubell, Jeffrey Flanigan, Andrew McCallum, and Elsa Olivetti. The materials science procedural text corpus: Annotating materials synthesis procedures with shallow semantic structures. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 56–64, 2019.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. Brat: A web-based tool for nlp-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, page 102–107, USA, 2012. Association for Computational Linguistics.

Ronen Tamari, Hiroyuki Shindo, Dafna Shahaf, and Yuji Matsumoto. Playing by the book: An interactive game approach for action graph extraction from text. In *Proceedings of the Workshop on Extracting Structured Knowledge from Scientific Publications*, pages 62–71, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-2609. URL https://www.aclweb.org/anthology/W19-2609.

Alain C. Vaucher, Federico Zipoli, Joppe Geluykens, Vishnu H. Nair, Philippe Schwaller, and Teodoro Laino. Automated extraction of chemical synthesis actions from experimental procedures. *Nature Communications*, 11(1):1–11, 2020. ISSN 20411723. doi: 10.1038/s41467-020-17266-6. URL http://dx.doi.org/10.1038/s41467-020-17266-6.

# A Annotation Schema

In the following subsections, we provide details of the annotation schema used: the PEG node and edge type ontology, along with examples and the rules governing the annotation process.

## A.1 Nodes

### A.1.1 Predicates (Operations)

Operation nodes correspond to "action" entities in WLP. In X-WLP, to facilitate conversion to executable instructions, we further add a fine-grain operation type; for each operation, annotators were required to select the closest operation type, or a `general` type if none applied. Operation types are broadly aligned with Autoprotocol operation types, but are broader in scope, to not limit applicability to any one platform. For example, we use a more general `measure` operation type rather than the specific types of measurement operations in Autoprotocol (`spectrophotometry`, `measure-volume`, etc.). Table 1 lists operation types, their frequency in the data, and example mentions.

### A.1.2 Arguments (Entities)

Predicate arguments, including both physical entities such as reagents, containers and devices, as well as more abstract entities like measurements (*1 ml*) and operation settings (*30 minutes*). Table 2 lists entity types, their frequency in the data, and example mentions.

## A.2 Edges

In general, edges are represented by triplets of the form $(s, r, t)$ where $s$ and $t$ are argument nodes (§A.1.2) and $r$ is a core or non-core role. Dependent on a particular role $r$, certain restrictions may apply to the fine-grained type of $s$ and $t$, as described below.

### A.2.1 Core Roles

Core roles, displayed in Table 3, represent operation specific roles, for example "ARG1" for the `seal` operation is a *seal* entity representing the seal of the "ARG0" argument. For core roles, the following restrictions hold:

- Source nodes $s$ are restricted to any of the *object* types $s \in \{\textit{reagent}, \textit{device}, \textit{seal}, \textit{location}\}$ representing physical objects. The only exception to this rule is that "ARG1" for the `seal` operation must be a *seal* entity.
- Target node $t$ is a predicate of one of the types in §A.1.1.

Table 1: Details of PEG predicate types, along with example frequent trigger spans and relative frequency in X-WLP.

| Operation Type | Frequent example spans | Count | Pct. |
|---|---|---|---|
| Transfer | add, transfer, place | 1301 | 33.2 |
| Temperature Treatment | incubate, store, thaw | 503 | 12.8 |
| General | Initiate, run, do not vortex | 469 | 11.9 |
| Mix | mix, vortex, inverting | 346 | 8.8 |
| Spin | spin, centrifuge, pellet | 282 | 7.2 |
| Create | prepare, make, set up | 178 | 4.5 |
| Destroy | discar, decant, pour off | 170 | 4.3 |
| Remove | remove, elute, extract | 168 | 4.3 |
| Measure | count, weigh, measure | 149 | 3.8 |
| Wash | wash, rinse, clean | 146 | 3.7 |
| Time | wait, sit, leave | 114 | 2.9 |
| Seal | cover, seal, cap | 68 | 1.7 |
| Convert | change, transform, changes | 21 | 0.5 |

Table 2: Details of PEG argument types, along with example frequent trigger spans and relative frequency in X-WLP.

| Entity Type | Frequent example spans | Count | Pct. |
|---|---|---|---|
| Reagent | supernatant, dna, sample | 3362 | 32.6 |
| Measurement | 1.5 mL, 595nm, 1pmol | 1924 | 18.6 |
| Setting | overnight, room temperature | 1622 | 15.7 |
| Location | tube, ice, plates | 1373 | 13.3 |
| Modifier | genty, carefully, immediately | 1070 | 10.3 |
| Device | forceps, pipette tip, water bath | 590 | 5.7 |
| Method | dilutions, up and down, pipetting | 271 | 2.6 |
| Seal | lid, cap, aluminim foil | 97 | 0.9 |

Table 3: Details of core role semantics for all operation types. The "Required" column specifies which roles must be filled for a given operation. ARG* is short for $\{\text{ARG0}, \text{ARG1}, \text{ARG2}\}$.

| Operation | Role Semantics | Required |
|---|---|---|
| Spin | ARG0 centrifuged to produce solid phase ARG1 and/or liquid phase ARG2 | ARG0 |
| Convert | ARG0 converted to ARG1 | ARG0, ARG1 |
| Seal | ARG0 sealed with ARG1 | ARG0 |
| Create | ARG* to be measured | ARG0 |
| General | - | ARG0 |
| Destroy | ARG0 discarded | ARG0 |
| Measure | ARG* to be measured | ARG0 |
| Mix | ARG* are mixed | ARG0 |
| Remove | ARG0 removed from ARG1 | ARG0 |
| Temperature Treatment | ARG* to be heated/cooled | ARG0 |
| Time | Wait after operation on ARG0 | ARG0 |
| Transfer | ARG* are sources, transferred to "site" | ARG0, site |
| Wash | ARG0 washed with ARG1 | ARG0, ARG1 |

- $r$ is a core argument relation, $r \in \{\text{ARG0}, \text{ARG1}, \text{ARG2}\}$ or ARG* for short.
- Certain roles may be required for a valid predicate $t$, for example the `transfer` operation requires at minimum both source and target arguments to be specified by the ARG0 and site roles, respectively.

### A.2.2 Non-core Roles

Non-core roles (e.g., "setting", "site", or "co-ref") indicate predicate-agnostic labels. For example, the *site* argument always marks the location in which a predicate is taking place. Non-core roles are

Table 4: Details of non-core roles and restrictions on source and target node types. Object is short for the set of entity types representing physical objects: $\{\textit{reagent}, \textit{device}, \textit{seal}, \textit{location}\}$.

| Role | Source Types | Target Types |
|---|---|---|
| co-ref | Object | Object |
| measure | Measurement | Object |
| setting | Setting | Object |
| modifier | Modifier | Object, Operation, Measurement |
| usage | Method, Object | Operation |
| located-at | Object | Object |
| part-of | Object | Object |

displayed in Table 4, and role-specific restrictions on $s$ and $t$ are listed under "Source Types" and "Target Types", respectively.

# B Dataset Details

## B.1 Comparison with other datasets

Table 5 compares X-WLP with similar procedural text corpora in terms of average document, sentence and word counts. As can be seen, X-WLP is on par with other recent datasets, and annotates complex documents, constituting more than 15 sentences on average.

## B.2 Inter-annotator agreement

Similarly to our PEG representation, the AMR formalism has predicate and argument nodes (lab operations and entities in our notation) and directed labeled edges which can form undirected cycles through reentrancies (nodes with multiple incoming edges). In Table 6 we report a graph Smatch score [Cai and Knight, 2013] widely used to quantify AMR's graph structure agreement, as well as finer grained graph agreement metrics, adapted from Damonte et al. [2017]. Smatch values are comparable to those obtained for AMR, where reported gold agreement varies between $0.69-0.89$ [Cai and Knight, 2013], while our task deals with longer, paragraph length representations. Reentrancies are the hardest for annotators to agree on, probably since they involve longer-range, typically cross-sentence relations. On the other hand, local decisions such as argument and predicate identification achieve higher agreement, and also benefit greatly from the annotations of WLP.

## B.3 Reentrancies and cross-sentence relation statistics

Table 7 presents a breakdown of all relation types along with cross-sentence and re-entrancy information. As noted in §3, analysis of the relations in Table 7 reveals that a significant proportion of arguments in PEGs are re-entrancies (32.4%) or cross-sentence (50.3%). Note that for these calculations we consider only argument relations that can in principle occur as re-entrancies: "ARG*" and "site", see relation ontology in Appendix A.2 for details. The cross-sentence calculation includes co-reference closure information.

Table 5: Statistics of our annotated corpus (X-WLP), compared with the ProPara corpus [Dalvi et al., 2018], material science (MSPTC; Mysore et al. [2019]) and chemical synthesis procedures (CSP; Vaucher et al. [2020]). CSP is comprised of annotated sentences (document level information is not provided)

| | X-WLP (ours) | MSPTC | CSP | ProPara |
|---|---|---|---|---|
| # words | 54k | 56k | 45k | 29k |
| # words / sent. | 12.7 | 26 | 25.8 | 9 |
| # sentences | 4,262 | 2,113 | 1,764 | 3,300 |
| # sentences / docs. | 15.28 | 9 | N/A | 6.8 |
| # docs. | 279 | 230 | N/A | 488 |

Table 6: X-WLP inter-annotator agreement metrics. Smatch Cai and Knight [2013] quantifies overall graph structure. Following metrics provide a finer-grained break down Damonte et al. [2017].

| **Agreement Metric** | **F1** |
|---|---|
| Smatch | 84.99 |
| Argument identification | 89.72 |
| Predicate identification | 86.68 |
| Core roles | 80.52 |
| Re-entrancies | 73.12 |

Table 7: Breakdown of PEG relation types by frequency in X-WLP, showing counts of inter/intra-sentence relations. Re-entrancies are possible only for core and "site" arguments, and may be either inter or intra-sentence.

| Relation | # Intra. | # Inter. | Total | # Re-entrancy |
|---|---|---|---|---|
| Core | | | | |
| • ARG0 | 2962 | 952 | 3914 | 1645 |
| • ARG1 | 560 | 127 | 687 | 3 |
| • ARG2 | 84 | 123 | 207 | 77 |
| Total (core) | 3606 | 1202 | 4808 | 1725 |
| Non-Core | | | | |
| • site | 1306 | 325 | 1631 | 360 |
| • setting | 3499 | 2 | 3501 | - |
| • usage | 1114 | 24 | 1138 | - |
| • co-ref | 129 | 1575 | 1704 | - |
| • located-at | 199 | 72 | 271 | - |
| • measure | 2936 | 18 | 2954 | - |
| • modifier | 1861 | 2 | 1863 | - |
| • part-of | 72 | 65 | 137 | - |
| Total (non-core) | 11116 | 2083 | 13199 | 360 |
| Temporal | 1218 | 788 | 2006 | - |
| Grand Total | 15940 (80%) | 4073 (20%) | 20013 | 2085 |