Process-Level Representation of Scientific Protocols with Interactive Annotation

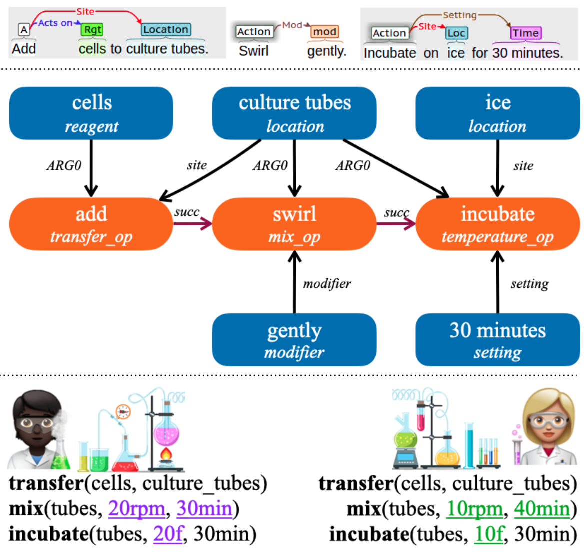
Ronen Tamari[†*] **Fan Bai**[‡] **Alan Ritter**[‡] **Gabriel Stanovsky**[†⋆]
[†]The Hebrew University of Jerusalem
[‡]Georgia Institute of Technology
[⋆]Allen Institute for Artificial Intelligence
`{ronent,gabis}@cs.huji.ac.il`
`{fan.bai,alan.ritter}@cc.gatech.edu`

Abstract

We develop Process Execution Graphs (PEG), a 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 manually annotate PEGs in a corpus of complex lab protocols with a novel interactive textual simulator that keeps track of entity traits and semantic constraints during annotation. We use this data to develop graph-prediction models, finding them to be good at entity identification and local relation extraction, while our corpus facilitates further exploration of challenging long-range relations.[1]

1 Introduction

There is a drive in recent years towards automating wet lab environments, where menial benchwork procedures such as pipetting, centrifuging, or incubation are software-controlled, and either executed by fully automatic lab equipment (Lee and Miles, 2018), or with a human-in-the-loop (Keller et al., 2019). 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; Prabhu and Urban, 2017). 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*.



transfer(cells, culture_tubes)
mix(tubes, 20rpm, 30min)
incubate(tubes, 20f, 30min)

transfer(cells, culture_tubes)
mix(tubes, 10rpm, 40min)
incubate(tubes, 10f, 30min)

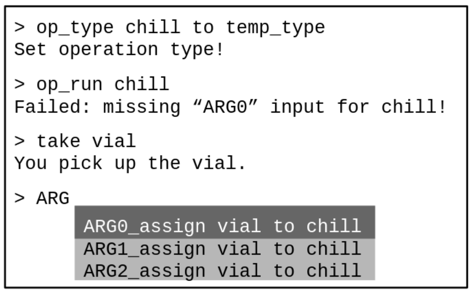
Figure 1: We develop a scaffold (center) between sentence-level lab procedure representations (top) and low-level, lab-specific instructions (bottom). The Process Execution Graph (PEG) captures document-level relations between procedures (orange rounded nodes) and their arguments (blue rectangular nodes).

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 instructions, ensuring a non-ambiguous, repeatable description of experiments.

In this work, we focus on a crucial first step towards such conversion – 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 temperature of the

*Work begun on an internship at the Allen Institute for Artificial Intelligence.

[1] Our annotated corpus, simulator, annotation interface, interaction data, and models are available for use by the research community at `https://textlabs.github.io/`.



```
> op_type chill to temp_type
Set operation type!

> op_run chill
Failed: missing "ARG0" input for chill!

> take vial
You pick up the vial.

> ARG
        ARG0_assign vial to chill
        ARG1_assign vial to chill
        ARG2_assign vial to chill
```

Figure 2: Example interaction with our simulator, showing predicate grounding ("chill" is a `temp_type` operation) input assignment ("vial" is an argument of "chill"), validation (warning for a missing argument) and auto-complete driven by state-tracking, where only legal instructions in a given state are presented.

incubation in Figure 1.

Formally, PEGs are directed, acyclic labeled graphs, 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 simulating objects and actions in a lab environment (see example in Figure 2). Our annotators are given wet-lab protocols written in natural language taken from biochemistry publications, 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 `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.

Finally, we use this data to explore several models, building upon recent advances in graph prediction algorithms (Luan et al., 2019; Wadden et al., 2019). We thoroughly analyze model performance and find that our data introduces interesting new challenges, such as complex co-reference resolution and long-range, cross-sentence relation identification.

In conclusion, we make the following contributions:

- We formalize a PEG representation for free-form, natural language lab protocols, providing a semantic scaffold between free-form scientific literature and low-level instruments instruction.
- 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 by non-experts.
- We release X-WLP, a challenging corpus of 279 PEGs representing document-level lab protocols. This size is on par with similar corpora of procedural text (Dalvi et al., 2018; Mysore et al., 2019; Vaucher et al., 2020).
- We develop two graph parsers: a pipeline model which chains predictions for graph sub-components, and a joint-model of mention and relation detectors.

2 Background and Motivation

Several formalisms for programmatic lab controller interfaces were developed in recent years (Yachie and Natsume, 2017; Lee and Miles, 2018). For instance, Autoprotocol defines 35 lab commands, including `spin`, `incubate`, and `mix`.[2] While these define wet-lab experiments in a precise and unambiguous manner, they do not readily replace their natural language description in scientific publications, much like a model implementation in python does not replace its high-level description in ML papers. Similarly to ML model descriptions, lab protocols are often not specified enough to support direct conversion to low-level programs. For example, the protocol in Figure 1 does not specify the swirling (mixing) speed or its duration.

Our process execution graph (PEG) captures the predicate-argument structure of the protocol, allowing it to be more lenient than a programming language (for example, capturing that *gently* modifies `swirl`). Better suited to represent underspecified natural language, PEGs can serve as a convenient scaffold to support downstream tasks such as text-to-code assistants (Mehr et al., 2020). For example, by asking researchers to fill in missing required arguments for `swirl`.

To annotate PEGs, we leverage the sentence-level annotations of Kulkarni et al. (2018) (WLP henceforth). WLP, exemplified at the top of

[2] https://autoprotocol.org/specification

Figure 1, collected sentence-level structures using the BRAT annotation tool (Stenetorp et al., 2012). For example, capturing that *cells*, *culture tubes* are arguments for `add`. However, WLP does not capture cross-sentence implicit relations such that *culture tubes* are an argument for `incubate`. These are abundant in lab protocols, require tracking entities across many sentences, and are not easy to annotate using BRAT (see discussion in §4). We vastly extend upon WLP annotations, aiming to capture the full set of expressed protocol relations, using a novel text-based games annotation interface which lends itself to procedural text annotation.

3 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 3, and formally defined below.

Nodes PEG nodes are triggered by explicit text spans in the protocol, e.g., “swirl", or “ice”. Nodes consist of two types: (1) *predicates*, marked in orange: denoting lab operations, such as `add` or `incubate`; 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*).

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

Argument 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	gently, carefully, clean	1070	10.3
Device	forceps, pipette tip	590	5.7
Method	dilutions, pipetting	271	2.6
Seal	lid, cap, aluminum foil	97	0.9

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

Node grounding The PEG formulation above is motivated as a scaffold towards fully-executable lab programs employed in automatic lab environments. To achieve this, we introduce an ontology for each of the node types, based on the Autoprotocol specification (Lee and Miles, 2018), as indicated below each text span in Figures 1 and 3. For example, `swirl` corresponds to an Autoprotocol `mix` operation, a *culture tube* is of type *location*, and *30 minutes* is a *setting*. See Tables 1, 2 for details of predicate and argument types respectively, their frequencies in our data and example spans.

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`. See Table 3 for predicate-specific core-role semantics, and Table 6 for non-cores roles types and frequencies of all roles in X-WLP. See Appendix A.3 for the rules defining what relations can hold between various entity types.

Relation to Autoprotocol As shown at the bottom of Figure 1, a PEG is readily convertible to Autoprotocol or similar laboratory interfaces once it is fully instantiated, thanks to edge labels and node grounding to an ontology. For example, a

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* are created	ARG0
General	-	ARG0
Destroy	ARG* 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

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 $\{\text{ARG0}, \text{ARG1}, \text{ARG2}\}$.

researcher can specify what *gently* means in terms of mixing speed for their particular lab instruments.

Reentrancies and cross-sentence relations While the PEG does not form directed cycles,[3] it does form non-directed cycles (or *reentrancies*) – where there exists nodes u, v such that there are two different paths from u to v. This occurs when an object participates in two or more temporally-dependent operations. For example, see *culture tubes*, which participates in all operations in Figure 1. In addition, 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. In the following section we will show that both reentrancies and cross-sentence relations, which are not captured by previous annotations, are abundant in our annotations.

4 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 (for details on ontology coverage, see §A.1).

In total, we collected 3,708 sentences (54.1K tokens) in 279 wet lab protocols annotated with our graph representation. As can be seen in Table 4, X-WLP annotates long examples, often spanning dozens of sentences, and its size is comparable (e.g., in terms of annotated words) to the ProPara corpus (Dalvi et al., 2018) and other related procedural datasets.

[3] This happens because the temporal relations define a partial ordering imposed by the linearity of the execution.

	X-WLP (ours)	MSPTC	CSP	ProPara
# words	54k	56k	45k	29k
# words / sent.	14.6	26	25.8	9
# sentences	3,708	2,113	1,764	3,300
# sentences / docs.	13.29	9	N/A	6.8
# docs.	279	230	N/A	488

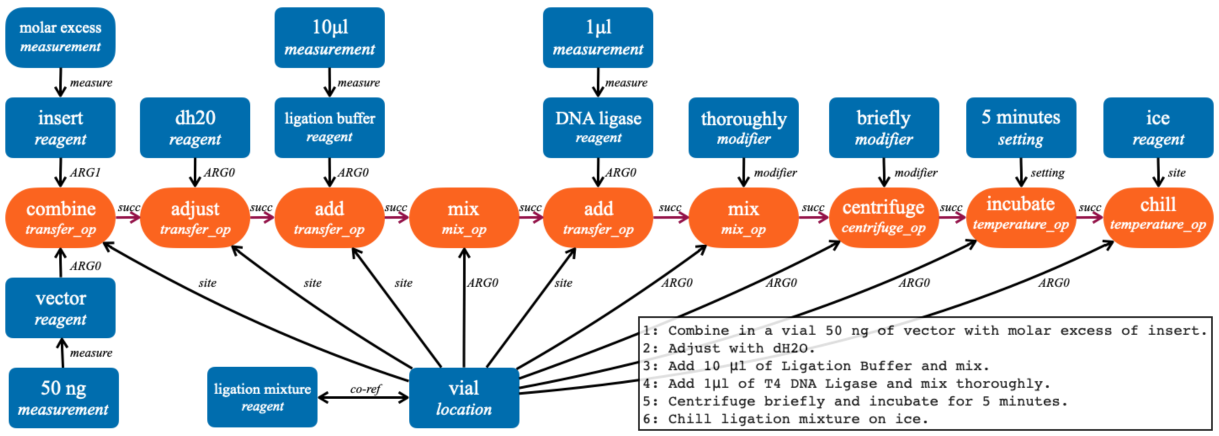
Table 4: Statistics of our annotated corpus (X-WLP). X-WLP annotates complex documents, constituting more than 13 sentences on average. X-WLP overall size is on par with other recent procedural corpora, including ProPara (Dalvi et al., 2018), material science (MSPTC; Mysore et al. (2019)) and chemical synthesis procedures (CSP; Vaucher et al. (2020)). CSP is comprised of annotated sentences (document level information is not provided).

4.1 WLP as a Starting Point

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 ask annotators to enrich them with type grounding for operations and arguments, as well as cross-sentence relations, as defined in §3. From these annotations we obtain process-level representations as presented in Figures 1 and 3.

4.2 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 3. 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 indeed, such complicated annotation has been a factor in the omission of cross-sentence information in similar domains (Mysore et al., 2019). A simulator can provide a natural way to account for it by



```
1: Combine in a vial 50 ng of vector with molar excess of insert.
2: Adjust with dH2O.
3: Add 10 µl of Ligation Buffer and mix.
4: Add 1µl of T4 DNA Ligase and mix thoroughly.
5: Centrifuge briefly and incubate for 5 minutes.
6: Chill ligation mixture on ice.
```

Figure 3: A full process gold PEG annotation from X-WLP for a real-world wet lab protocol whose text is presented in the lower right corner (protocol 512), exemplifying several common properties: (1) complex, technical 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*).

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 human annotator interacts with the text-based interface to simulate the raw wet lab protocol (Figure 2): setting the types of operations (the first interaction sets the span "chill" as a `temperature` operation) and assigning their inputs (the last line assigns `vial` as an input to `chill`), 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 2 indicates a missing argument for the `chill` operation (the argument to be chilled). 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 an auto-complete tool (last interaction in Figure 2), 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.

See the project web page for the complete annotation guidelines, visualizations of annotated protocols, and demonstration videos of the annotation process.

4.3 Data Analysis

Four in-house CS undergraduate students with interest in NLP used our simulator to annotate the protocols of X-WLP, where 44 of the protocols were annotated by two different annotators to estimate agreement.

Inter-annotator agreement. We turn to the literature on abstract meaning representation (AMR; Banarescu et al., 2013) for established graph agreement metrics, which we adapt to our setting. 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).[4] In Table 5 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

[4] Unfortunately, we cannot follow this analogy to train AMR models on our graphs, since, to the best of our knowledge, they are currently limited to single sentences, notwithstanding a promising recent initial exploration into multi-sentence AMR annotation (O'Gorman et al., 2018).

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

Table 5: 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).

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

Table 6: 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.

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.

Information gain from process-level annotation. Analysis of the relations in X-WLP, presented in Table 6, reveals that a significant proportion of arguments in PEGs are re-entrancies (32.4%) or cross-sentence (50.3%).[5] Figure 3 shows a representative example, with the *vial* participating in multiple re-entrancies and long-range relations,

[5] 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.3 for details. Cross-sentence calculation includes co-reference closure information.

Dataset	Avg. #args/op	#Ops. w/o core arg.	#Ops.	Pct.
WLP	1.87	3297	17485	18.9
X-WLP	3.01	0	3915	0.0

Table 7: Comparison of average arguments per operation and percentage of semantically under-specified operations (missing core arguments) in WLP and X-WLP.

triggered by each sentence in the protocol. These relations are crucial to correctly model the protocols at the process level, and are inherently missed by sentence-level formalisms, showing the value of our annotations.

To shed light on the additional process-level information captured by our approach relative to WLP, in Table 7 we compare the average number of arguments per operation node as well as the amount of operation nodes with no core arguments. 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.

Annotation cost. The time to annotate an average document of 13.29 sentences was approximately 53 minutes (roughly 4 minutes per sentence), not including annotator training. Our annotator pay was 13 USD / hour. The overall annotation budget for X-WLP was roughly 3,200 USD.

5 Models

We present two approaches for PEG prediction. First, in §5.1 we design models for separate graph sub-component prediction, which are chained to form a pipeline PEG prediction model. Second, in §5.2 we present a model which directly predicts the entire PEG using a span-graph prediction approach.

5.1 Pipeline Model (PIPELINE)

A full PEG representation as defined in §3 can be obtained by chaining the following models which predict its sub-components. In all of these, we use SciBERT (Beltagy et al., 2019) which was trained on scientific texts similar to our domain.

Mention identification. Given a scientific protocol written in natural language, we begin by identifying all experiment-involved text spans mentioning lab operations (predicates) or entities and their traits (arguments), which are the building blocks for PEGs. We model this problem of mention identification as a sequence tagging problem. Specifically, we transfer span-level mention labels, which are annotated in the WLP corpus into token-level labels using the BIO tagging scheme, then fine-tune the SciBERT model for token classification.

Predicate grounding. Next, we ground predicate nodes into the operation ontology types discussed in §3. See Table 1 in the Appendix for the complete list. Predicted mentions are marked using special start and end tokens (`[E-start]` and `[E-end]`), then fed as input to SciBERT. The contextual embedding of `[E-start]` is input to a linear softmax layer to predict the fine-grained operation type.

Operation argument role labeling. Once the operation type is identified, we predict its semantic arguments and their roles. Given an operation and an argument mention, four special tokens are used to specify the positions of their spans (Baldini Soares et al., 2019). Type information is also encoded into the tokens, for example, when the types of the operator and its argument are `mix-op` and `reagent` respectively, four special tokens `[E1-mix-op-start]`, `[E1-mix-op-end]`, `[E2-rg-start]` and `[E2-rg-end]` are used to denote the spans of the mention pair. After feeding the input into SciBERT, the contextualized embeddings of `[E1-op-mix-start]` and `[E2-rg-start]` are concatenated as input to a linear layer that is used to predict the entity's argument role. Arguments of an operation can be selected from anywhere in the protocol, leading to many cross-sentence operation-argument link candidates. To accommodate cross-sentence argument roles, we use the entire document as input to SciBERT for each mention pair. However, SciBERT is limited to processing sequences of at most 512 tokens. To address this limitation, longer documents are truncated in a way that preserves surrounding context, when encoding mention pairs.[6] Only 8 of the 279 protocols in our dataset contain more than 512 tokens.

Temporal ordering. Finally, we model order of operations using the `succ` relation (see Figure 3). These are predicted using a similar approach as argument role labeling, where special tokens are used to encode operation spans.

5.2 Jointly-Trained Model (MULTI-TASK)

To explore the benefits of jointly modeling mentions and relations, we experiment with a graph-based multi-task framework based on DYGIE++ model (Wadden et al., 2019). Candidate mention spans are encoded using SciBERT, and a graph is constructed based on predicted X-WLP relations and argument roles. A message-passing neural network is then used to predict mention spans while propagating information about related spans in the graph (Dai et al., 2016; Gilmer et al., 2017; Jin et al., 2018).

This approach requires computing hidden state representations for all $O(n^4)$ pairs of spans in an input text, which for long sequences, will exhaust GPU memory. While Wadden et al. (2019) considered primarily within-sentence relations, our model must consider relations across the entire protocol, which makes this a problem of practical concern. To address this, we encode a sliding window of w adjacent sentences when the full protocol does not fit into memory, allowing smaller windows for the start and end of the protocol, and concatenate sentences within each window as inputs to the model. As a result, each sentence is involved in w windows leading to repeated, possibly contradicting predictions for both mentions and relations. To handle this, we output predictions agreed upon by at least k windows, where k is a hyperparameter tuned on a development set.

6 Experiments

In §5, we presented a pipelined approach to PEG prediction based on SciBERT and a message-passing neural network that jointly learns span and relation representations. Next, we describe the details of our experiments and present empirical results demonstrating that X-WLP supports training models that can predict PEGs from natural language instructions.

[6] Given an input document, which has more than 512 words, with n words between two mentions, we truncate the context to keep at most $(512 - n)/2$ words for each side.

Data Split	System	F_1
original	Kulkarni et al. (2018)	78.0
	Wadden et al. (2019)	**79.7**
	PIPELINE	78.3
X-WLP-eval	PIPELINE	74.7

Table 8: Mention identification test set F_1 scores for models on the WLP dataset. Top: WLP dataset with the original train/dev/test split. Bottom: excluding X-WLP protocols from the WLP training data, and using them for evaluation.

Data. X-WLP is our main dataset including 279 fully annotated protocols. Statistics of X-WLP are presented in Table 4. Additionally, we have 344 protocols from the original WLP dataset. We use this auxiliary data only for training mention taggers in the pipeline model, and use X-WLP for all other tasks. For argument role labeling and temporal ordering, negative instances are generated by enumerating all possible mention pairs whose types appear at least once in the gold data. We use 5-fold cross validation; 2 folds (112 protocols) are used for development, and the other 3 folds (167 protocols) are used to report final results.

Model setup. The PIPELINE framework employs a separate model for each task, by default using the propagated predictions from previous tasks as input. In addition, we evaluate the model for each task with gold input denoted as PIPELINE `(gold)`. Finally, the MULTI-TASK framework learns all tasks together and we decompose its performance into the component subtasks.

Implementation details. We use the uncased version of SciBERT[7] for all our models due to the importance of in-domain pre-training. The models under the PIPELINE system are implemented using Huggingface Transformers (Wolf et al., 2020), and we use AdamW with the learning rate 2×10^{-5} for SciBERT finetuning. For the MULTI-TASK framework, we set the widow size w to 5, the maximum value that enables the model to fit in GPU memory. For all other hyperparameters, we follow the settings of the WLP experiments in (Wadden et al., 2019).

6.1 Results

The results of the two models on the different subtasks are presented in Tables 8- 11. We identify three main observations based on these results.

[7] `https://github.com/allenai/scibert`

System	P	R	F_1
MULTI-TASK	76.0	69.0	72.3
PIPELINE	71.8	76.3	74.0
• w/ gold mentions	79.0	80.2	79.6

Table 9: Predicate grounding test set results.

Task	MULTI-TASK	PIPELINE	# gold
Core			
• All roles	**57.9**	53.7	2839
• All roles (gold mentions)	-	76.5	2839
• ARG0	**61.0**	57.1	2313
• ARG1	**36.1**	32.9	412
• ARG2	**69.7**	61.4	114
Non-Core			
• All roles	**55.7**	48.8	4826
• All roles (gold mentions)	-	78.1	4826
• site	**58.7**	55.4	962
• setting	**77.4**	74.7	974
• usage	**35.6**	33.0	296
• co-ref	**39.8**	36.7	1014
• measure	**63.3**	56.6	804
• modifier	**51.0**	41.8	519
• located-at	9.7	**13.3**	179
• part-of	0.5	**10.8**	78
Temporal Ordering	**61.8**	57.3	2176
Temp. Ord. (gold mentions)	-	76.3	2176

Table 10: Operation argument role labeling (core and non-core roles, decomposed by relation) and temporal ordering test set F_1 performance.

Split	MULTI-TASK	PIPELINE	# gold
Intra-sentence	**63.4**	58.2	2160
Inter-sentence	**32.5**	39.1	679

Table 11: Operation argument role labeling (core roles) test set F_1, decomposed based on whether the operation and the argument are triggered within the same sentence (intra-sentence) versus different sentences (inter-sentence).

First, PIPELINE outperforms MULTI-TASK on the operation classification task in Table 9, as it uses all protocols from WLP as additional training data to improve mention tagging.

Second, MULTI-TASK performs better than the PIPELINE approach on most relation classification tasks in Table 10, but is worse than PIPELINE when PIPELINE uses gold mentions, demonstrating that jointly modeling mentions and relations helps in mitigating error propagation.

Third, *cross-sentence* relations are challenging for both models, as shown in Table 11. This explains the low performance of `co-ref`, which is comprised of 92.4% cross-sentence relations.

In addition, there are a couple of interesting points to note. In Table 8, the performance of PIPELINE on the X-WLP subset is lower than its performance on the WLP test set, likely because there are fewer protocols in the training set. For the relation-decomposed performance in Table 10, we can see that some of the relations like "ARG2" can be correctly predicted by MULTI-TASK using only a few gold labels while some more widely used relations are harder to learn, such as "ARG0" and "site"; indeed, "ARG2" is only used in the `spin` operation (see Table 3), while the other roles participate in more diverse contexts.

7 Related Work

Natural Language Processing (NLP) for scientific procedural text is a rapidly growing field. To-date, most approaches have focused on text-mining applications (Isayev, 2019) and typically annotate only shallow, sentence-level semantic structures (e.g., Fig. 1, top). Examples include WLP (Kulkarni et al., 2018) and materials science procedures (Mysore et al., 2019; Kuniyoshi et al., 2020). Recent interest in automation of lab procedures has also led to sentence-level annotation of procedural texts with action sequences designed to facilitate execution (Vaucher et al., 2020).

However, as noted in recent concurrent work (Mehr et al., 2020), neither sentence-level semantic structures nor action sequences are sufficient for the goal of converting text to a machine-executable synthesis procedure; for this purpose, a more structured, process-level semantic representation is required. In particular, executable representations require a structured declaration of the locations and states of the different materials throughout a process, details not represented by sentence-level annotations. Our simulator can naturally represent such information by maintaining a stateful model of the process. Simulation fidelity can be controlled by implementing the execution semantics of operations to the level of detail required.

Mehr et al. (2020) have similarly proposed a process-level executable representation, but use an NLP pipeline consisting primarily of rules and simple pattern matching, relying on a human-in-the-loop for corrections; linking our approach with their framework is a promising future direction.

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 and predicting procedure-level representations.[8]

Another line of research focuses on procedural text understanding for more general domains: simple scientific processes (Dalvi et al., 2018), open domain procedural texts (Tandon et al., 2020), and cooking recipes (Kiddon et al., 2015; Bosselut et al., 2018). These works represent process-level information and entity state changes, but typically feature shorter processes, simpler language and an open ontology, compared with our domain-specific terminology and grounded ontology.

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)) towards *natural* language understanding.

8 Conclusion

We developed a novel meaning representation and simulation-based annotation interface, enabling the collection of process-level annotations of experimental procedures, as well as two parsers (pipeline and joint modelling) trained on this data. Our dataset and experiments present several directions for future work, including the modelling of challenging long range dependencies, application of text-based games for procedural text understanding, and extending simulation-based annotation to new domains.

Acknowledgments

We would like to thank Peter Clark, Noah Smith, Yoav Goldberg, Dafna Shahaf, and Reut Tsarfaty for many fruitful discussions and helpful comments, as well as the X-WLP annotators: Pranay Methuku, Rider Osentoski, Noah Zhang and Michael Zhan. This work was partially supported by an Allen Institute for AI Research Gift to Gabriel Stanovsky. This material is based upon work supported by the NSF (IIS-1845670) and the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001119C0108. The views, opinions, and/or findings expressed are those of the author(s) and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

[8] In addition, in contrast with AMR, PEG nodes are directly mapped to the trigger spans in the document.

References

Ashutosh Adhikari, Xingdi Yuan, Marc-Alexandre Côté, Mikuláš Zelinka, Marc-Antoine Rondeau, Romain Laroche, Pascal Poupart, Jian Tang, Adam Trischler, and Will Hamilton. 2020. Learning dynamic belief graphs to generalize on text-based games. *Advances in Neural Information Processing Systems*, 33.

Livio Baldini Soares, Nicholas FitzGerald, Jeffrey Ling, and Tom Kwiatkowski. 2019. Matching the blanks: Distributional similarity for relation learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2895–2905, Florence, Italy. Association for Computational Linguistics.

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

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

Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China. Association for Computational Linguistics.

Antoine Bosselut, Corin Ennis, Omer Levy, Ari Holtzman, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. In *International Conference on Learning Representations*.

Shu Cai and Kevin Knight. 2013. 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. Association for Computational Linguistics.

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

Hanjun Dai, Bo Dai, and Le Song. 2016. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pages 2702–2711.

Bhavana Dalvi, Lifu Huang, Niket Tandon, Wen-tau Yih, and Peter Clark. 2018. 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. Association for Computational Linguistics.

Marco Damonte, Shay B. Cohen, and Giorgio Satta. 2017. 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. Association for Computational Linguistics.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org.

Olexandr Isayev. 2019. Text mining facilitates materials discovery. *Nature*, 571(7763):42–43.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2018. Junction tree variational autoencoder for molecular graph generation. In *International Conference on Machine Learning*.

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

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

Ben Keller, Justin Vrana, Abraham Miller, Garrett Newman, and Eric Klavins. 2019. Aquarium: The Laboratory Operating System version 2.6.0.

Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. 2015. 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. Association for Computational Linguistics.

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

Chaitanya Kulkarni, Wei Xu, Alan Ritter, and Raghu Machiraju. 2018. 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. Association for Computational Linguistics.

Fusataka Kuniyoshi, Kohei Makino, Jun Ozawa, and Makoto Miwa. 2020. Annotating and extracting synthesis process of all-solid-state batteries from scientific literature. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 1941–1950, Marseille, France. European Language Resources Association.

Peter L Lee and Benjamin N Miles. 2018. 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.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3036–3046, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Ben Miles and Peter L. Lee. 2018. Achieving reproducibility and closed-loop automation in biological experimentation with an iot-enabled lab of the future. *SLAS TECHNOLOGY: Translating Life Sciences Innovation*, 23(5):432–439. PMID: 30045649.

Sheshera Mysore, Zachary Jensen, Edward Kim, Kevin Huang, Haw-Shiuan Chang, Emma Strubell, Jeffrey Flanigan, Andrew McCallum, and Elsa Olivetti. 2019. 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.

Tim O'Gorman, Michael Regan, Kira Griffitt, Ulf Hermjakob, Kevin Knight, and Martha Palmer. 2018. AMR beyond the sentence: the multi-sentence AMR corpus. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3693–3702, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Gurpur Rakesh D Prabhu and Pawel L Urban. 2017. The dawn of unmanned analytical laboratories. *TrAC Trends in Analytical Chemistry*, 88:41–52.

Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. 2012. 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. Association for Computational Linguistics.

Ronen Tamari, Hiroyuki Shindo, Dafna Shahaf, and Yuji Matsumoto. 2019. 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. Association for Computational Linguistics.

Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi, Dheeraj Rajagopal, Peter Clark, Michal Guerquin, Kyle Richardson, and Eduard Hovy. 2020. A dataset for tracking entities in open domain procedural text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6408–6417, Online. Association for Computational Linguistics.

Alain C. Vaucher, Federico Zipoli, Joppe Geluykens, Vishnu H. Nair, Philippe Schwaller, and Teodoro Laino. 2020. Automated extraction of chemical synthesis actions from experimental procedures. *Nature Communications*, 11(1):1–11.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5784–5789, Hong Kong, China. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Nozomu Yachie and Tohru Natsume. 2017. Robotic crowd biology with maholo labdroids. *Nature Biotechnology*, 35(4):310–312.

A Annotation Schema

In the following subsections, we provide further details of the annotation schema used. Section §A.1 describes how the ontology was constructed based on Autoprotocol, and §A.2 provides details on ontology coverage for the X-WLP protocols which were chosen for annotation. Section §A.3 details the rules defining valid PEG edges, or what relations can hold between various entity types. The annotation guidelines given to annotators are available on the project web page.

A.1 Ontology Construction

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.

To define our operation type ontology, we consulted the Autoprotocol (Miles and Lee, 2018) open source standard used for executable biology lab protocols. Autoprotocol defines 35 different operation types,[9] from which we grouped relevant types into higher level clusters; X-WLP operation types are broadly aligned with Autoprotocol operation types, but are more general 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 12 maps between X-WLP operation types and their equivalents in Autoprotocol, if one exists. The X-WLP operation types do not perfectly overlap with Autoprotocol as the former is written for humans, while the latter is designed for the more constrained domain of robot execution. Accordingly, some operations not currently supported in Autoprotocol were added, like `wash`. See Table 1 for example mention spans for each X-WLP operation type.

The set of supported operations was chosen to maximize coverage over the types of operations found in the sentence-level annotations of WLP (see §A.2 below for details).

[9] Based on `https://github.com/autoprotocol/autoprotocol-python/blob/master/autoprotocol/instruction.py` as of January 2021.

A.2 Ontology Coverage

To identify candidate protocols for annotation which were well covered by the ontology, we created a mapping between ontology instruction types and the 100 most frequent text-spans of WLP action entities (constituting 74% of all action spans in WLP). WLP action text spans that didn't correspond to any ontology instruction were mapped to a `general` label; action text spans that could be mapped to the ontology we call ontology-covered actions. For annotation in X-WLP, we then selected WLP protocols estimated to have a high percentage of ontology-covered actions (based on the mapping above). This simple method was found to be effective in practice, as measured by the actual ontology coverage of X-WLP annotations, summarized in Fig. 4.

For each annotated protocol, we calculated the percentage of known (not `general`) operations. Fig. 4 plots, for each coverage percentile (y-axis), the percentage (x-axis) of X-WLP protocols with at least y percent known operations. From the plot we can see for example that half of the protocols in X-WLP have >90% ontology coverage, and 90% of the protocols have >70% ontology coverage.

X-WLP Operation	Autoprotocol Instructions
Spin	Spin
Convert	N/A
Seal	Seal, Cover
Create	Oligosynthesize, Provision
General	N/A
Destroy	N/A
Measure	Absorbance, Fluorescence, Luminescence, IlluminaSeq, SangerSeq, MeasureConcentration, MeasureMass, MeasureVolume, CountCells, Spectrophotometry, FlowCytometry, FlowAnalyze, ImagePlate
Mix	Agitate
Remove	Unseal, Uncover
Temperature Treatment	Thermocycle, Incubate, FlashFreeze
Transfer	AcousticTransfer, MagneticTransfer, Dispense, Provision, LiquidHandle, Autopick
Wash	N/A
Time	N/A

Table 12: Mapping between X-WLP operation types and corresponding Autoprotocol instructions (if any exist). Autoprotocol operations tend to be more specific as they are intended for machine execution. X-WLP protocols are written for humans, so operation types are defined at a higher level of abstraction.

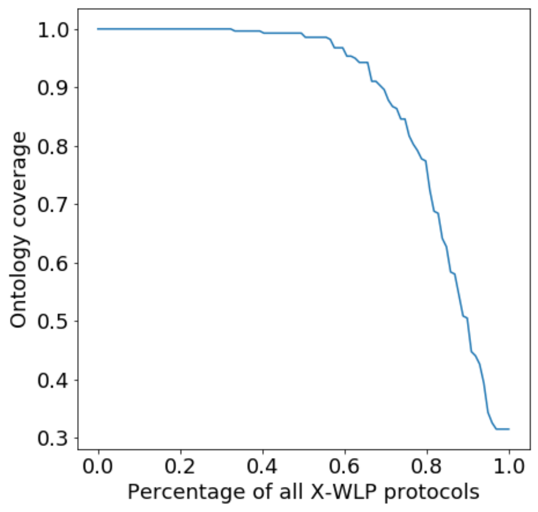


Figure 4: Plot displaying for each coverage percentile (y-axis), the percentage (x-axis) of X-WLP protocols with at least y percent known (ontology-covered) operations.

A.3 Syntax governing PEG edges

Formally, edges are represented by triplets of the form (s, r, t) where s and t are argument nodes 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.3.1 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 $s \in \{reagent, device, seal, location\}$ 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 Table 1.
- 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.

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

Table 13: 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: $\{reagent, device, seal, location\}$.

A.3.2 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 displayed in Table 13, and role-specific restrictions on s and t are listed under "Source Types" and "Target Types", respectively.