Annotation Instructions

Motivation

What are we doing?

In this project we are attempting to create a new kind of dataset to improve procedural text understanding algorithms. Procedural texts are natural language texts describing step-by-step instructions or recipes. Specifically, we'll focus on biology laboratory instructions, also called wet labs protocols.

Why are we doing it?

1. Most existing datasets for procedural texts don't provide detailed enough instructions/training environments such as required for an AI agent learning to execute a procedure.
2. To provide an easier interface between humans and instruction-following AI agents. Instead of having to specify instructions in some machine language code, the human operator could use something closer to natural language.

3. A dataset like this would be a rich and unique source of training data from the perspective of Natural Language Understanding (NLU) researchers - long, real world texts grounded to machine readable formats are quite rare.

How are we doing it?

Rather than building a 3D simulator of a laboratory to train agents, which would involve a lot of engineering effort, we'll work with TextLabs, a simple text-based simulator of a laboratory. This will let us focus purely on the language aspect. TextLabs provides an environment and set of instructions that can be used to interact with it. To further streamline the dataset construction, we will use the existing Wet Labs Protocols (WLP) dataset, and convert it to TextLabs format. WLP provides extensive annotations for ~600 protocols. We'll start with a shorter and simpler subset of this dataset.

How To Annotate

The annotations provided in WLP are sentence-level semantic parses, also known as action-graphs. Each parse is a graph over word spans (nodes) with various relations between them.

Such graphs are not machine executable, so we will convert them into executable instructions in the text-based simulator, as follows:

Each **typed node** in the WLP action graph is automatically converted to a **typed entity** in the game. These are the entities you'll be able to interact with.

Relations in the graph correspond roughly to **commands / instructions / actions** in the game.

There are 3 main entity types:

1. `Operation` (corresponding to WLP `Action`).
2. `Object`: (corresponding to WLP `Object` entities).
3. `Descriptor`: (corresponding to WLP `Measure-based` and `Parts-of-Speech based` entity types).

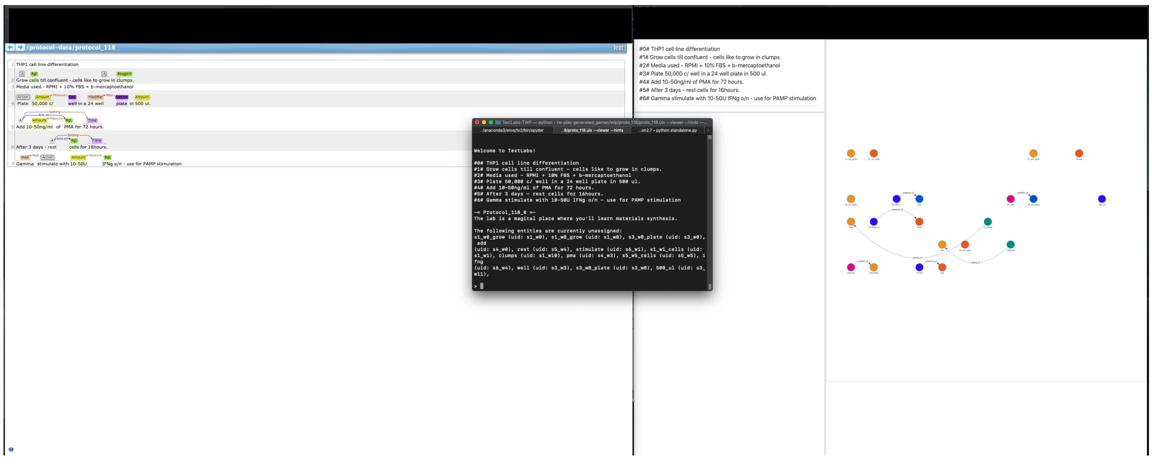
There are two main types of actions possible in TextLabs (roughly corresponding to relations in WLP):

1. Description Actions: connecting between `Descriptor` and `Object, Operation` entities. These are "shallow" actions which do not affect game state, beyond simple linkage of two entities.
2. Operation Events (corresponding to WLP action relation): we allow *typing* and *execution* of `Operation` entities, which can affect game state based on their type and arguments. This comprises a key difference with WLP, where actions are untyped (and non-executable of course). This is intended to align with the notion of executable instructions corresponding with those we may wish for an automated agent to perform.

If this sounds complicated, referring to the examples below can help clarify.

Suggested Annotation Flow

1. For a given WLP protocol, open it in brat alongside the TextLabs visualization screen. See the screenshot below for our annotation setup.



#0# THP1 cell line differentiation
#1# Grow cells till confluent - cells like to grow in clumps.
#2# Media used - RPMI + 10% FBS + b-mercaptoethanol
#3# Plate 50,000 c/ well in a 24 well plate in 500 ul.
#4# Add 10-50ng/ml of PMA for 72 hours.
#5# After 3 days - rest cells for 16hours.
#6# Gamma stimulate with 10-50U IFNg o/n - use for PAMP stimulation

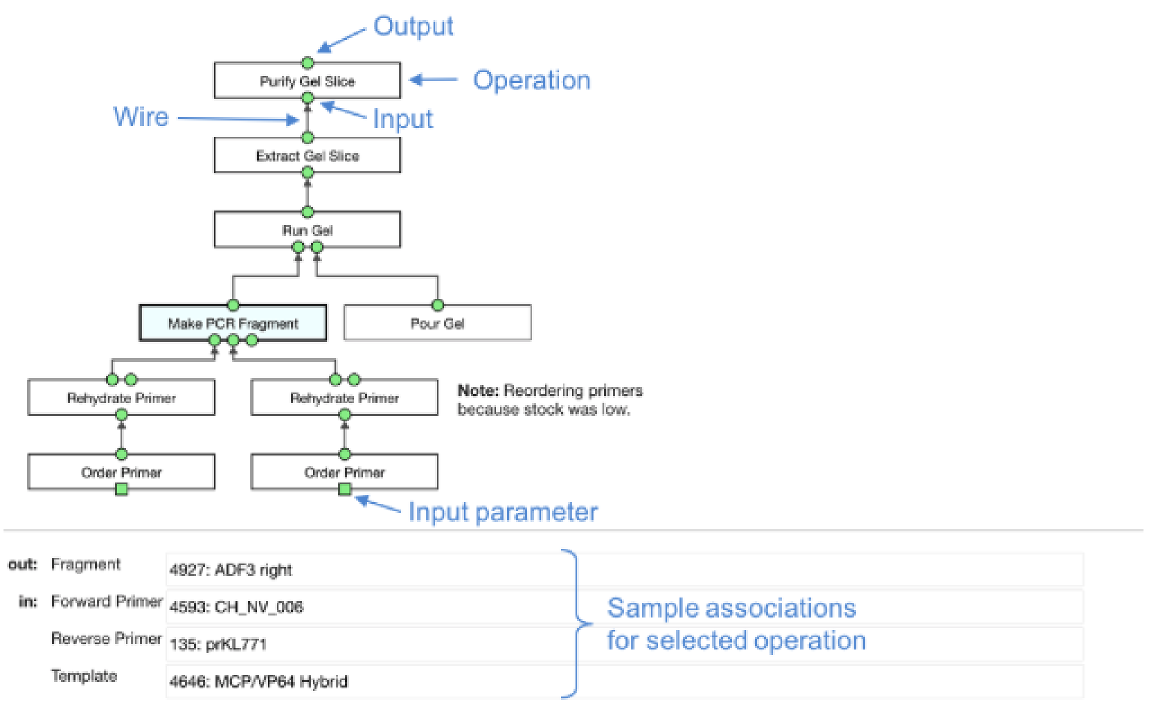
```
Welcome to TextLabs!

#0# THP1 cell line differentiation
#1# Grow cells till confluent - cells like to grow in clumps.
#2# Media used - RPMI + 10% FBS + b-mercaptoethanol
#3# Plate 50,000 c/ well in a 24 well plate in 500 ul.
#4# Add 10-50ng/ml of PMA for 72 hours.
#5# After 3 days - rest cells for 16hours.
#6# Gamma stimulate with 10-50U IFNg o/n - use for PAMP stimulation

-= Protocol_118_0 =-
The lab is a magical place where you'll learn materials synthesis.

The following entities are currently unassigned:
s1_w0_grow (uid: s1_w0), s1_w8_grow (uid: s1_w8), s3_w0_plate (uid: s3_w0),
 add
(uid: s4_w0), rest (uid: s5_w4), stimulate (uid: s6_w1), s1_w1_cells (uid:
s1_w1), clumps (uid: s1_w10), pma (uid: s4_w3), s5_w5_cells (uid: s5_w5), i
fng
(uid: s6_w4), well (uid: s3_w3), s3_w8_plate (uid: s3_w8), 500_ul (uid: s3_
w11),

> 
```

brat
/protocol-data/protocol_118
TextLabs-TWP — python • tw-play generated_games/wlp/proto_118/proto_118.ulx --viewer --hints —...
.../anaconda3/envs/tx2/bin/spyder
...8/proto_118.ulx --viewer --hints
...on2.7 • python standalone.py

2. Proceed from the top sentence, for each sentence:
   1. Try to understand the type of operation\s (if any) in the sentence, and set accordingly.
   2. Check that the preloaded relations seem to make sense, and if not- undo them (simply by `taking` the source of the relation).
   3. Prepare the operation's input slots (`a`, `b`, `c`, `site`) as needed, and then run the operation.
   4. Set co-reference between mentions of the same entity, with the earlier reference as the source and later reference as target. Co-reference will often span between sentences.

A correct annotation should typically lead to a **connected** executable action graph, meaning that all participant entities should be linked together by operation or description relations. The is not currently the case as the WLP dataset annotations span within but not across sentence.

To give you an idea of the structures we're trying to capture, this is an example of a real process graph as designed by a biology researcher. As can be seen, the routing between inputs and outputs are key to understanding overall process structure. Each of the blocks in the diagram below may correspond to a sentence or more in the WLP protocols dataset.



Note: Reordering primers because stock was low.

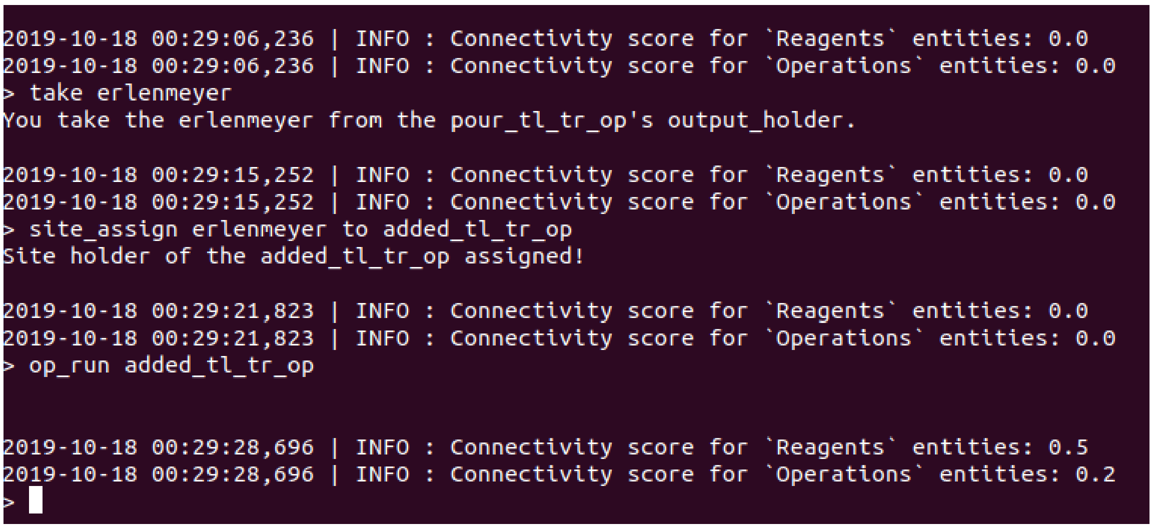
out:	Fragment	4927: ADF3 right
in:	Forward Primer	4593: CH_NV_006
	Reverse Primer	135: prKL771
	Template	4646: MCP/VP64 Hybrid

From: Aquarium Biology Lab Operating System

3. New in 1.0.5! During the annotation, after each command, you will be able to see connectivity scores for your annotations thus far. This is intended to help you reach a quality annotation which doesn't omit the key events, connections between them, and reagents (or if you do omit them, you should use `ignore`). Note how in the example above, the event graph is connected. Specifically, the simulator checks the key `Reagent` and

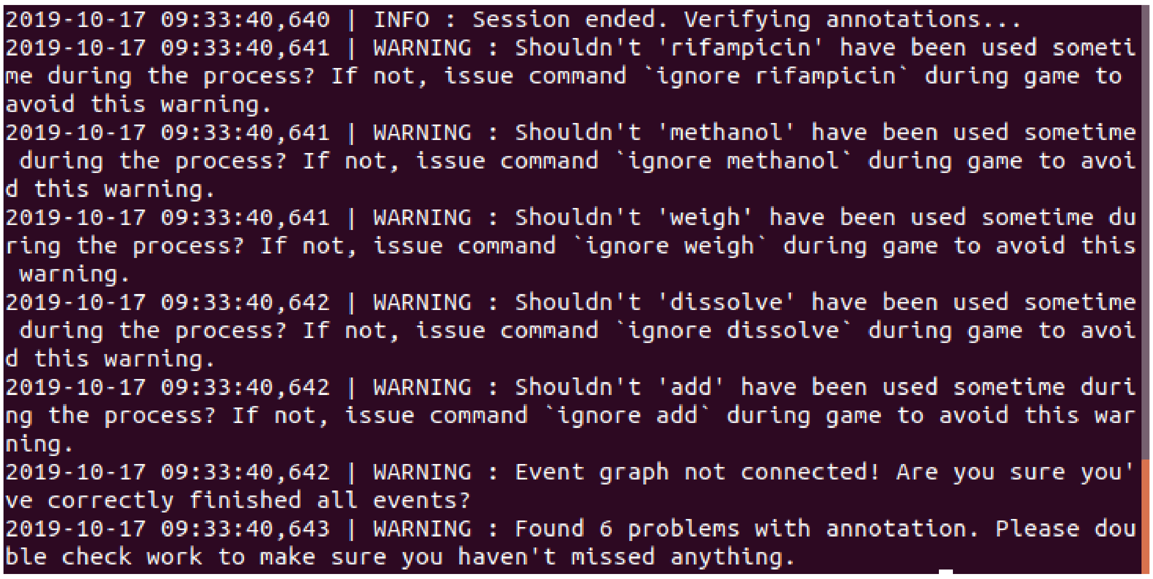
`Operation` entities to see if they are utilized and connected by the instructions. **The scores should increase as you add relations, and when you have finished the last sentence they should both be 1.0.**

Below is an example showing the score increasing as entities and events are connected.



```
2019-10-18 00:29:06,236 | INFO : Connectivity score for `Reagents` entities: 0.0
2019-10-18 00:29:06,236 | INFO : Connectivity score for `Operations` entities: 0.0
> take erlenmeyer
You take the erlenmeyer from the pour_tl_tr_op's output_holder.

2019-10-18 00:29:15,252 | INFO : Connectivity score for `Reagents` entities: 0.0
2019-10-18 00:29:15,252 | INFO : Connectivity score for `Operations` entities: 0.0
> site_assign erlenmeyer to added_tl_tr_op
Site holder of the added_tl_tr_op assigned!

2019-10-18 00:29:21,823 | INFO : Connectivity score for `Reagents` entities: 0.0
2019-10-18 00:29:21,823 | INFO : Connectivity score for `Operations` entities: 0.0
> op_run added_tl_tr_op


2019-10-18 00:29:28,696 | INFO : Connectivity score for `Reagents` entities: 0.5
2019-10-18 00:29:28,696 | INFO : Connectivity score for `Operations` entities: 0.2
>
```

4. At the end of the annotation, you will receive an automatic report verifying the annotations. If you have any warnings, please fix them to correct the annotation. If your scores are 1.0, you won't get any warnings.



```
2019-10-17 09:33:40,640 | INFO : Session ended. Verifying annotations...
2019-10-17 09:33:40,641 | WARNING : Shouldn't 'rifampicin' have been used someti
me during the process? If not, issue command `ignore rifampicin` during game to
avoid this warning.
2019-10-17 09:33:40,641 | WARNING : Shouldn't 'methanol' have been used sometime
 during the process? If not, issue command `ignore methanol` during game to avoi
d this warning.
2019-10-17 09:33:40,641 | WARNING : Shouldn't 'weigh' have been used sometime du
ring the process? If not, issue command `ignore weigh` during game to avoid this
 warning.
2019-10-17 09:33:40,642 | WARNING : Shouldn't 'dissolve' have been used sometime
 during the process? If not, issue command `ignore dissolve` during game to avoi
d this warning.
2019-10-17 09:33:40,642 | WARNING : Shouldn't 'add' have been used sometime duri
ng the process? If not, issue command `ignore add` during game to avoid this war
ning.
2019-10-17 09:33:40,642 | WARNING : Event graph not connected! Are you sure you'
ve correctly finished all events?
2019-10-17 09:33:40,643 | WARNING : Found 6 problems with annotation. Please dou
ble check work to make sure you haven't missed anything.
```

Note that if you have warnings, it means the annotation should be fixed. However, **if you don't have warnings- it still doesn't necessarily mean the annotation is perfect**- if we knew how to make perfect annotations, we wouldn't be needing the help of humans :) So meanwhile, you're the expert, try to make sure that your annotations make sense even if there were no warnings!

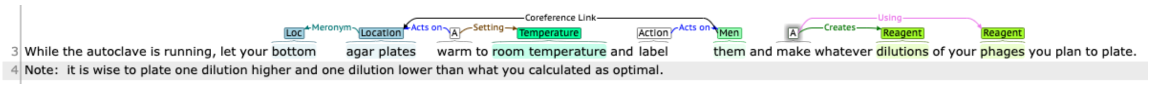
5. **Saving your work:** Note that `.txt` , `.peg` and `.tln` log files will be created where you specified using the `anno_log_dir` flag. This is what you should upload to `{google_drive_folder}/{protocol_number}` after you finish annotation.
6. Mark the protocol as complete under your column on your annotations spreadsheet- change from "T" (todo) to "F" (finished). Also, please fill in the time and summary scores as printed at the end of the session.

Only issue commands that appear in the autocomplete prompt. This is since Inform7 may allow commands that the Python side doesn't - issuing such a command would put the Python based state-tracking out of sync.

Notes (Important!)

- In a given protocol, multiple entities may have the same name, which can cause disambiguation problems for the command parser. To disambiguate mentions and allow the player to refer to a unique id if needed, a short unique id (of the form `s{n}_w{n}`, where `s`, `w` are the sentence and word numbers, respectively) is provided in parentheses and can always be used instead of the string description. Also, if an entity appears more than once, each mention will be accompanied by the sentence and word location: `s1_w2_seawater_sample` and the entity can be referred to by `s1_w2` if needed.
- Not every WLP relation or entity needs to be used (obviously for those not supported, but even some actions may be superfluous). In the example below, the "make dilutions" operation is not a machine executable command (it's more high level, for the human performing the protocol) and doesn't translate well into machine executable form. A rule of thumb to

help decide is that commands we want to annotate will tend to be simpler and closer to machine-executable form. If they aren't- consider leaving them out (see below).



3 While the autoclave is running, let your bottom agar plates warm to room temperature and label them and make whatever dilutions of your phages you plan to plate.
4 Note: it is wise to plate one dilution higher and one dilution lower than what you calculated as optimal.

- **Undoing Actions:** Changes since version 1.0.3.
  - To leave out an unused entity or action `E`, enter the command `ignore E`.
  - Note that every action you take (except Utility Actions) will be recorded as an Event for record-keeping purposes. For any action that creates an Event (Operations are Events by default, so no new events are created by running them), you can undo the effects as before, but in addition you must also `ignore` the created Event.
  - The preloaded relations are not associated with events, so can be undone as before.

Reference

TextLabs Entity Set

1. Objects
   - Reagent
   - Device
   - Location
   - Seal
2. Operations
3. Descriptors
   - Method
   - Modifier
   - Setting
   - Measurement

The full table mapping between WLP and TextLabs entities:

Wet Labs	TextLabs
action	operation
amount	measurement
concentration	measurement
device	device
generic-measure	measurement
location	location
measure-type	measurement
mention	none
method	method
misc	none
modifier	modifier
numerical	none
ph	measurement
reagent	reagent
seal	seal
size	measurement
speed	setting
temperature	setting
time	setting
unit	measurement

Action Set

Description Actions

These actions are best thought of as "plugging" a source entity into a target entity, thus creating a new description relation. A target entity, depending on its type, has a "bank of sockets" for each relation type it can be connected with. A source entity has single "plug type" for each relation type. For example, an `operation` entity has "sockets" for the `setting` relation, and the `measurement` entity has a `setting` relation "plug".

1. **Coreference**.

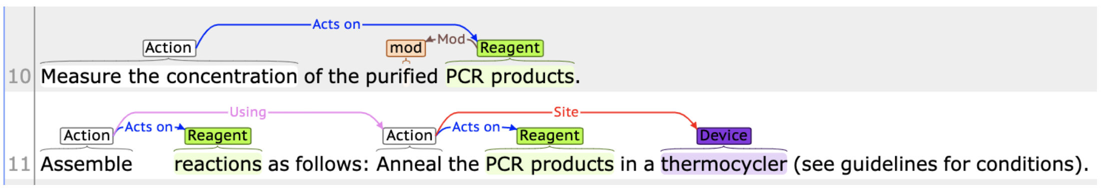
   Definition:

A link that associates two phrases when those two phrases in a text refer to the same entity.

Syntax:

```
co_ref {reagent, location, device} to {reagent, location, device}
```

Examples:



10 Measure the concentration of the purified PCR products.

11 Assemble reactions as follows: Anneal the PCR products in a thermocycler (see guidelines for conditions).

Informal Guidelines

- When running `co_ref a to b` , the active entity will remain `b`, not `a`!
- For consistency, **link from the source to the newest mention**. In the example above, `a` should be the `PCR products` entity in sentence 10, and `b` should be the one in sentence 11.
- Unlike in WLP schema, a co-reference link need not be contained in the span of a sentence, and may cross sentences. In the example below, clearly `PCR products` are the same entity in both sentences and should be consolidated.

2. **Modifier link**

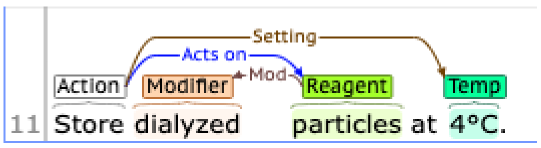
Definition:

Link modifier to entity it is attempting to modify.

Syntax:

```
mod_link {modifier} to {reagent, location, device, seal, operation, setting, measurement}
```

Examples:



11 Store dialyzed particles at 4°C.

3. **Settings link**

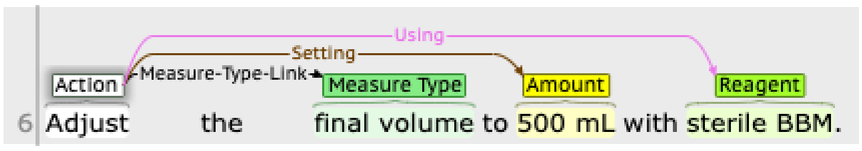
Syntax:

`setting_link {setting, measurement} to {operation}`

Definition:

Links devices or tools to their settings directly, if there is no action word associated with making those settings.

Examples:



4. **Measure link**

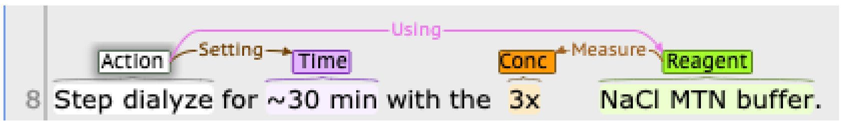
Syntax:

`set_measure {measurement} to {reagent, location, device, seal}`

Definition:

A link that associates the various numerical measures to the entity it is trying to measure.

Examples:



5. **Use linking**

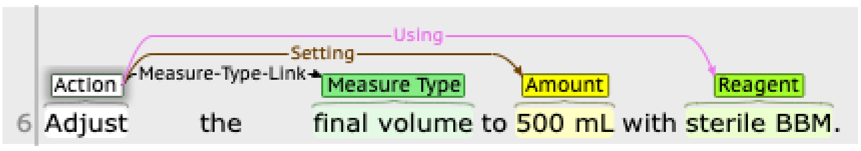
Syntax:

`use_link {reagent, location, device, seal, method} to {operation}`

Definition:

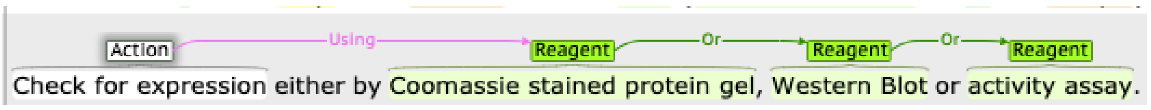
Any entity that the action verb makes 'use' of is linked with this relation. Any entity that
the action verb utilizes to perform the action

Examples:

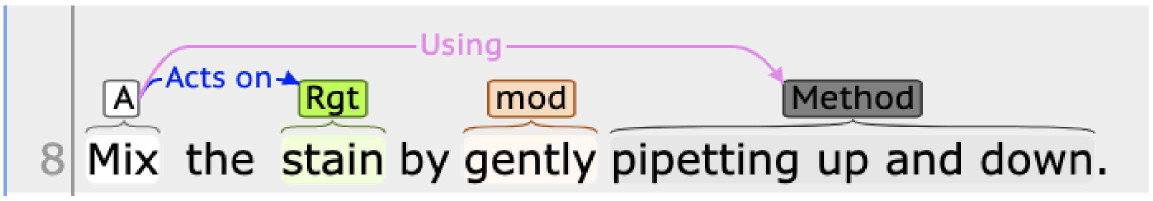


Informal Guidelines

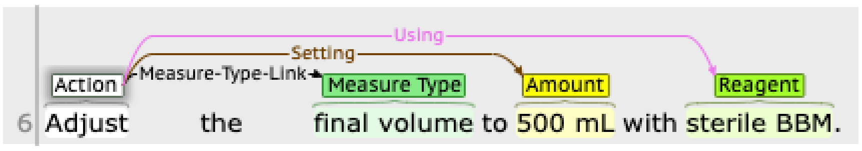
- Note that although this operation is similar to the `using` relation in WLP, we intend it to work differently. In TextLabs, this relation should only be used when the entity being used is **not** affecting the world state in ways that we track (primarily, composition and location of entities). An example to demonstrate the difference: in this case, the entity being used (the `stained protein gel`) does not affect the state in ways that we track, it only serves as part of the measurement procedure (doesn't alter composition or location of entities being measured).


Action
Using
Reagent
Or
Reagent
Or
Reagent
Check for expression either by Coomassie stained protein gel, Western Blot or activity assay.

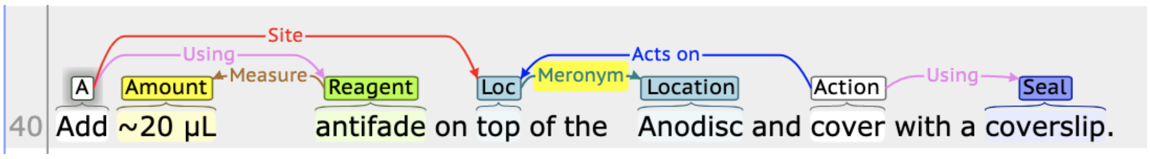
Also in this case, the use_link relation should be used as the pipetting isn't altering the operation in any way that we measure:



While in this case the entity being used (`sterile BBM`) is actually altering the composition since it's being added to some mixture, so we would use the `site_assign` command to express this.



Here as well, we do track the `seal`



40 Add ~20 µL antifade on top of the Anodisc and cover with a coverslip.

6. Part-of link

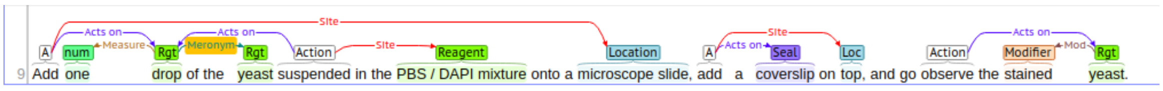
Syntax:

```
assemble {reagent, location, device, seal} to {reagent, location, device, seal}
```

Definition:

Setting some reagent, location or device entity to be a part of another reagent, location or device.

Examples:

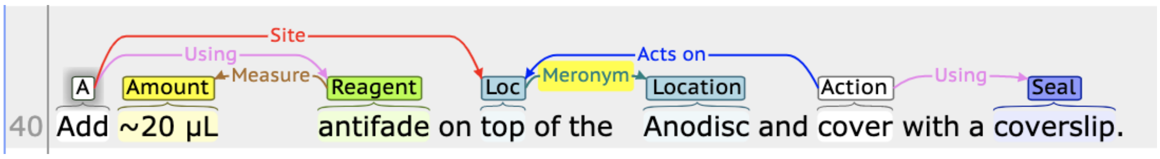


9 Add one drop of the yeast suspended in the PBS / DAPI mixture onto a microscope slide, add a coverslip on top, and go observe the stained yeast.

In the case above, you would perform `assemble drop to yeast`, and following that you would take the drop and input it to the `Add` operation as usual

Informal Guidelines:

The meronym relation doesn't always map to the Part-of link! It often will also be more appropriate to use the locate action. A rule of thumb is that 2 seperate entities (often one *contained* in another) should be related by the `locate` action, while an entity which is *part of* another (such as the `top` of a `container` should be related by the assemble action), see below:



40 Add ~20 µL antifade on top of the Anodisc and cover with a coverslip.

💡 **Existing Relations**: As description relations have only a limited affect on game state, we automatically generate most of these from the brat annotations, to reduce workload. Note that they can be modified, and indeed in some cases should be, to correct/improve existing annotations (primarily for the using relation).

Operation Actions

Operations in TextLabs can be thought of as **functions** with **types**, **inputs**, **effects** and **outputs**. In this preliminary stage, we support up to 3 `Object` entity input arguments and one `Site` argument.

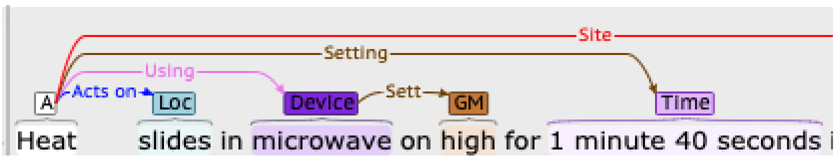
Input: arguments may be required or optional, depending on operation type.

Type: An operation's type can be set during run time, this must be done before any inputs are assigned. See below for details regarding types.

Running operations: see below. Effects depend on type of operation, and include moving, destroying and creating new relations between entities.

Output: After running the operation, outputs may or may not be created, to see what was created you can check the visualization or perform `x {operation}` , which will list all current operation details including inputs/outputs.

1. Input Assignment (similar to `Acts-on` relation in WLP)
    - Syntax: `input_{a, b, c}_assign {reagent, location, device, seal} to {operation}` (depending on the input slot a/b/c you wish to assign).
    - In most cases, arguments are positional (position sensitive), for example, in the `convert` operation, input a is converted into input b.
2. Site assigning (similar to `site` relation in WLP)
    - Syntax: `site_assign {reagent, location, device, seal} to {operation}`
    - After running the operation, if a site has been assigned, the operation outputs will be relocated to the site. Can be required or optional.
    - For example, consider the following event:



Heat slides in microwave on high for 1 minute 40 seconds

The `Heat` entity is of type `temp_type`, and since the `microwave` is an active participant in terms of location, it should be added as the site of the Heat operation by the command `site_assign microwave to Heat`.

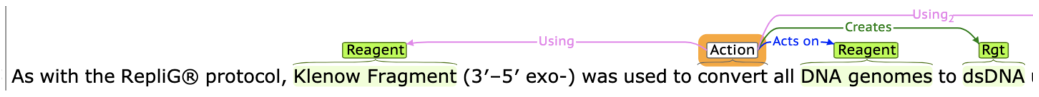
3. Set operation type
   - Syntax: `op_type {operation} to {op_type}`
   - Must set op type before assigning inputs.
   - Unless noted otherwise, all slots can be assigned with one of the following entity types: `{reagent, location, device, seal}`
   - Currently supported `op_type`s:
     1. `convert_type`

        Description:

        Can be used in cases where one entity is converted to another, typically through some reaction. Input `a` is the source entity and will be destroyed as a result of running the operation, `b` is the created entity. Represented meaning is "convert `a` to `b`")

        Example:

        `DNA genomes` → input `a`, `dsDNA` → input `b`,

        As with the RepliG® protocol, Klenow Fragment (3′–5′ exo-) was used to convert all DNA genomes to dsDNA

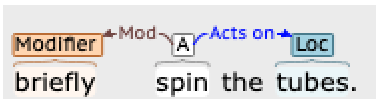
     2. `centrifuge_type`

        Description:

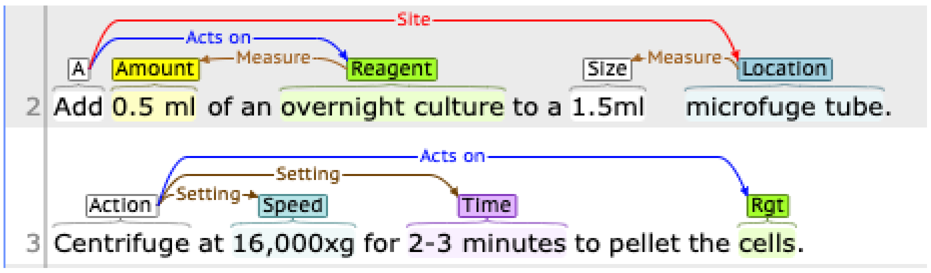
        Centrifuging refers to a special treatment where materials are reacted by spinning at high speed. This often creates pellets (solid) and supernatant (liquid) which can play active roles in procedure. We represent this by optionally allocating input `b` to the solid and input `c` to the liquid.

        Example:

In this case, input `a` ← `tubes`. No solids/liquids are mentioned.

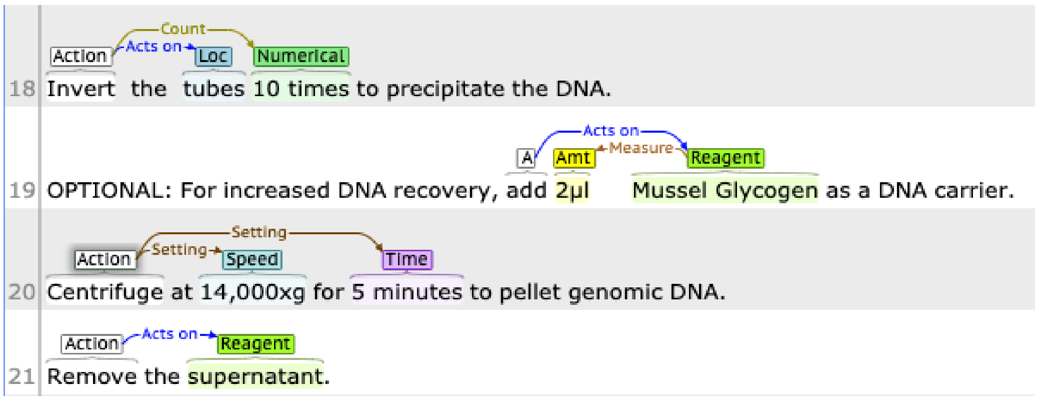

briefly spin the tubes.

In this case, input `a` ← `microfuge tube`, input `b` ← `cells`.


Add 0.5 ml of an overnight culture to a 1.5ml microfuge tube.

Centrifuge at 16,000xg for 2-3 minutes to pellet the cells.

In this case, input `a` ← `tubes`, input `b` ← `genomic DNA` (unfortunately not possible specifically here since the entity wasn't marked), input `c` ← `supernatant`.


Invert the tubes 10 times to precipitate the DNA.

OPTIONAL: For increased DNA recovery, add 2μl Mussel Glycogen as a DNA carrier.

Centrifuge at 14,000xg for 5 minutes to pellet genomic DNA.

Remove the supernatant.

3. `cover_type`

   Description:

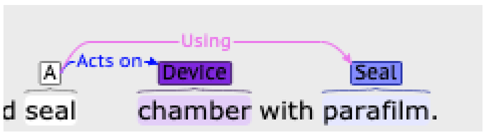
   Represents input slot `a` being covered/sealed by input `b` (which is optional, as could be unmentioned in text)

   Slots:

   - a: `{reagent, location, device}`
   - b: `{seal}`

   Example:

In this case, input `a` ← `chamber`, input `b` ← `parafilm`.

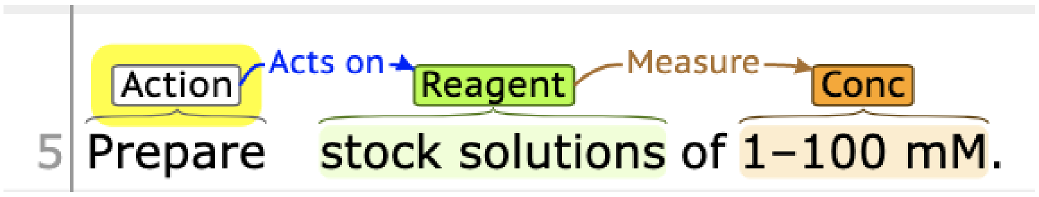


4. `create_type`

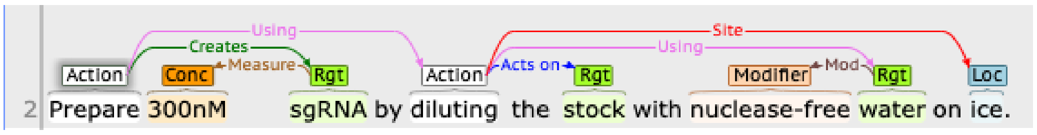
   Description:

   Represents an entity at input slot `a` being created, optionally can be used with input slots `b` as arguments representing representing the entity from *which* argument `a` was created (the semantics of this form are essentially "create `a` by doing `b`" or. Note that if argument `b` is supplied, it will be co-referenced to `a` as a result of running the operation.

   Example:
   An example with only input `a`, no `b`.



   An example with input `a` ← `sgRNA`, input `b` ← `stock`.

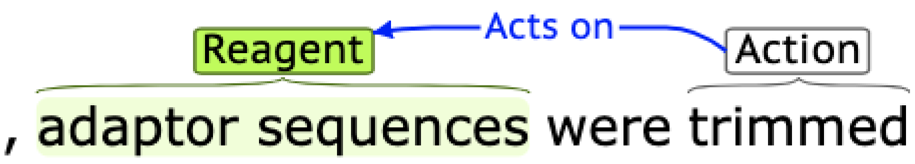

Using
Creates
Measure
Site
Using
Acts on
Mod
Action
Conc
Rgt
Action
Rgt
Modifier
Rgt
Loc
2 Prepare 300nM sgRNA by diluting the stock with nuclease-free water on ice.

5. `default_type`

   Description:

   This operation doesn't do anything to the inputs and should be used whenever you want to annotate an operation participating in the protocol that doesn't fit any of the other types. Obviously, try to keep use of this to a minimum :)

   Example:

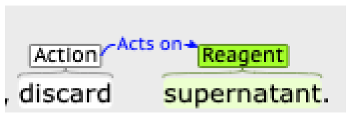


6. `destroy_type`

   Description:

   Represents input being discarded, after which it won't be used anymore in the process. Will affect state by removing the entity from play.

   Example:

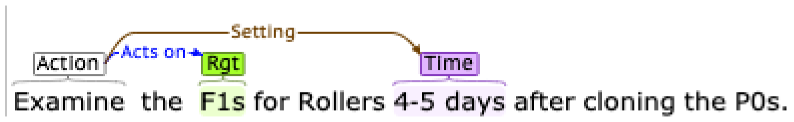


7. `measure_type`

   Description:

   Represents input being measured, doesn't affect state.

   Example:

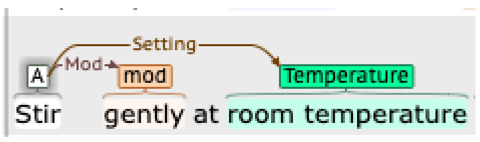


8. `mix_type`

   Description:

   Represents input being mixed, doesn't affect state.

   Example:



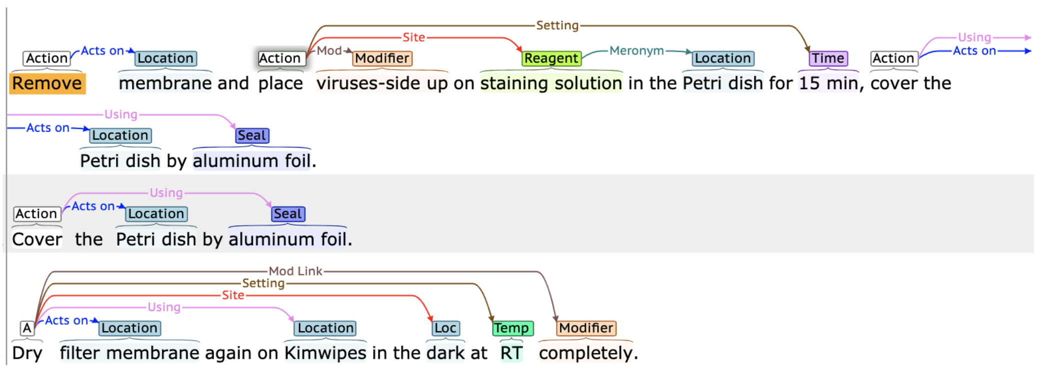
9. `remove_type`

   Description:

Context similar to `destroy_type` operation, but the `remove` operation doesn't affect state- use it in cases

1. Where an entity should be removed from a mixture, but not removed from the game since it will be needed later.
2. Where only part of an entity is removed, for example a certain volume of liquid.

Example:


Remove membrane and place viruses-side up on staining solution in the Petri dish for 15 min, cover the Petri dish by aluminum foil.

Cover the Petri dish by aluminum foil.

Dry filter membrane again on Kimwipes in the dark at RT completely.

In the example above, the membrane should be removed and then used again aftre 3 sentences. (case 1)

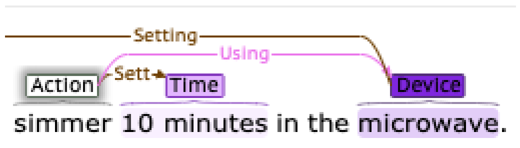
An example of case two could be a sentence like "Remove 200ml of liquid from tube."

10. `temp_type`

Description:

Represents input being heated/cooled/incubated/frozen, doesn't affect state.

Example:

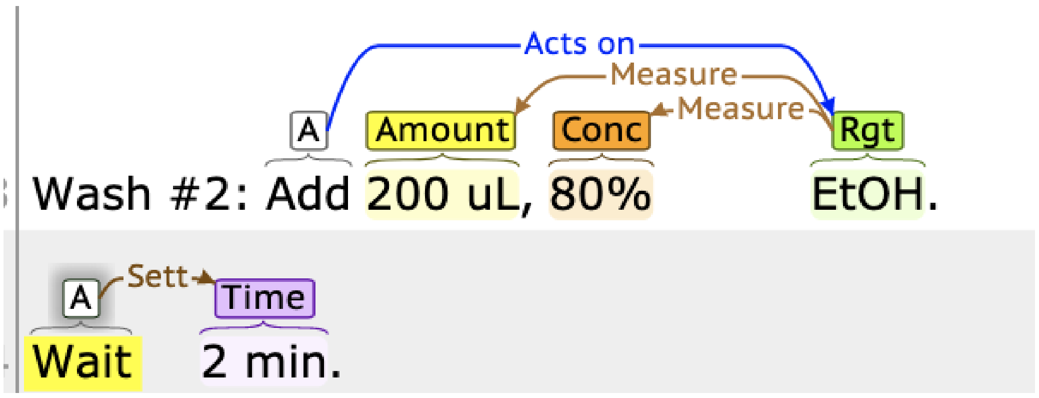

simmer 10 minutes in the microwave.

11. `time_type`

Description:

Represents an operation related to time, such as waiting., doesn't affect state.

Example:


Wash #2: Add 200 uL, 80% EtOH.

Wait 2 min.

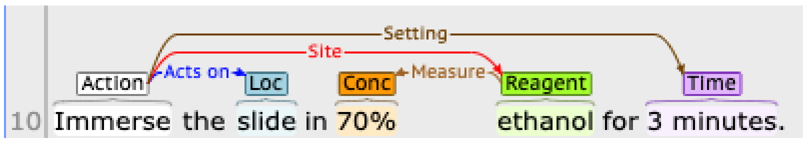
Note that the input argument to an operation like `Wait` argument may not be directly clear, but it is usually the output of the previous operation. In the example above, this would be the mixture to which `EtOH` was added.

12. `transfer_type`

    Description:

    Represents inputs being transferred to a new location. This is a very common operation. All occupied input slots `a`, `b` and `c` will be transferred to the entity at the `site` slot.

    Example:

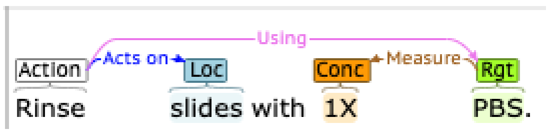

10 Immerse the slide in 70% ethanol for 3 minutes.

13. `wash_type`

    Description:

    Represents inputs `a` being washed with input `b` (`b` and `c` are optional. If `c` is added, this should be another input being washed by `b`).

    Example:


Rinse slides with 1X PBS.

`slides` → input `a`, PBS → input `b`.

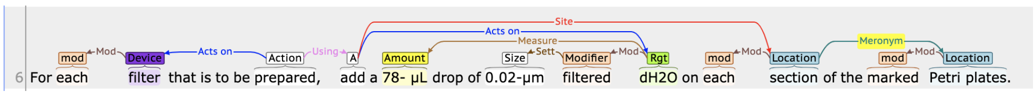
4. Operation running
   - Syntax: `op_run {operation}`
   - Must set op type and properly assign inputs before running. Running the operation will cause the state changes to take effect, according to action type.

Running operations cannot currently be undone

Implicit Actions

This allows executing actions not explicitly tied to an entity in the text, for example performing certain Operation Actions directly without needing an `Operation` entity.

1. Locate
   - Syntax: `locate {reagent, location, device, seal} to {reagent, location, device, seal}`
   - Move object to desired location, equivalent to what a `site` setting would do.
   - Only use this if no suitable `operation` entity is available! The meronym relation will often correspond to this action, as shown in the example below.
   - Example:



For each filter that is to be prepared, add a 78- µL drop of 0.02-µm filtered dH2O on each section of the marked Petri plates.

In this example, we would use the command `locate section to Petri plates`

TextLabs Utility Actions

1. **Take**.

   Definition:

This action is necessary before interacting with any entity, you can think of it as adding an entity to your "carried inventory". Currently the semantics for taking an entity involved in a binary relation are a little tricky - if you take the source entity of the relation, you will take only it and cancel the relation. If you take the target (socket) side, the relation will stay intact.

Syntax:

`take {object, descriptor}`

💡 All Description actions can be undone simply by `take`ing the source of the relation: for example `set_measure X to Y` followed by `take X`

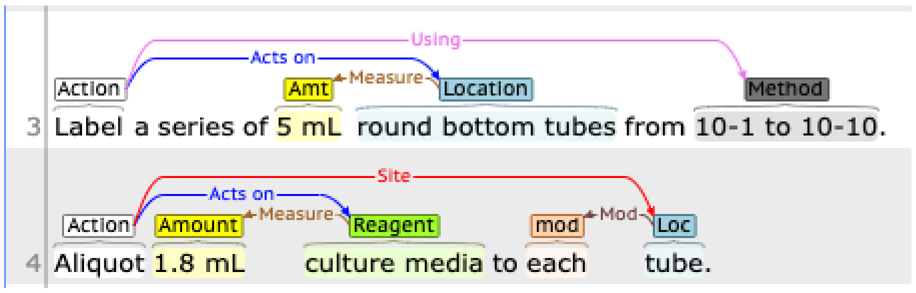
2. **Examine**
   - **Syntax**: `x {any entity}`
   - Examine an entity, yielding a description of its current state.

Limitations

- Natural language is notoriously difficult to capture in symbolic notation- some protocols or parts thereof may not be possible to express in the current version of TextLabs. Common examples are listed below. Please note cases of this while performing annotation (you can refer to them by protocol number + sentence number), this will greatly help in improving future versions.
- Some relations in WLP aren't yet supported in TextLabs.
- Unannotated entities cannot currently be interacted with.
- TextLabs operations currently support up to 3 inputs, some WLP actions may have more- just use as many as you can.
- Numerical reasoning is similarly not supported, so currently `Numerical` entities are unsupported (we could support them at the basic descriptor level where they would have no effect on game state).

Unsupported examples

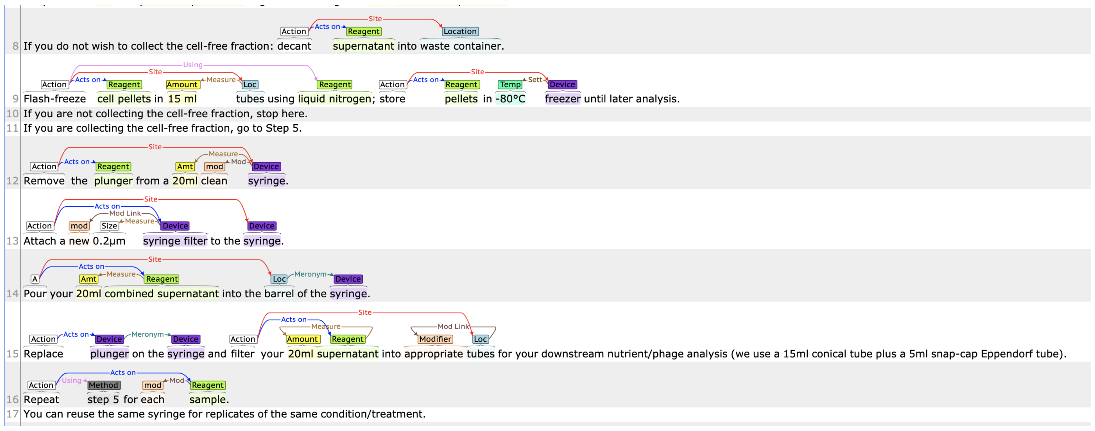
- Quantifiers such as `each`.



3 Label a series of 5 mL round bottom tubes from 10-1 to 10-10.

4 Aliquot 1.8 mL culture media to each tube.

- Conditions:



8 If you do not wish to collect the cell-free fraction: decant supernatant into waste container.
9 Flash-freeze cell pellets in 15 ml tubes using liquid nitrogen; store pellets in -80°C freezer until later analysis.
10 If you are not collecting the cell-free fraction, stop here.
11 If you are collecting the cell-free fraction, go to Step 5.
12 Remove the plunger from a 20ml clean syringe.
13 Attach a new 0.2µm syringe filter to the syringe.
14 Pour your 20ml combined supernatant into the barrel of the syringe.
15 Replace plunger on the syringe and filter your 20ml supernatant into appropriate tubes for your downstream nutrient/phage analysis (we use a 15ml conical tube plus a 5ml snap-cap Eppendorf tube).
16 Repeat step 5 for each sample.
17 You can reuse the same syringe for replicates of the same condition/treatment.