SchemaBlocks Annotation Guide (v0.1)

Introduction

The goal of this annotation project will be to hand annotate ***schematic*** knowledge into a machine readable format usable by automatic knowledge extraction and reasoning systems. Schematic knowledge (sometimes also called *script* knowledge) is a particular strain of common sense knowledge that deals with information about common scenarios one encounters in the actual world. A prototypical example of "common scenario" that one may have schematic knowledge about is the scenario of *going to a restaurant*. At a high level, schematic knowledge encodes two aspects about common sense scenarios:

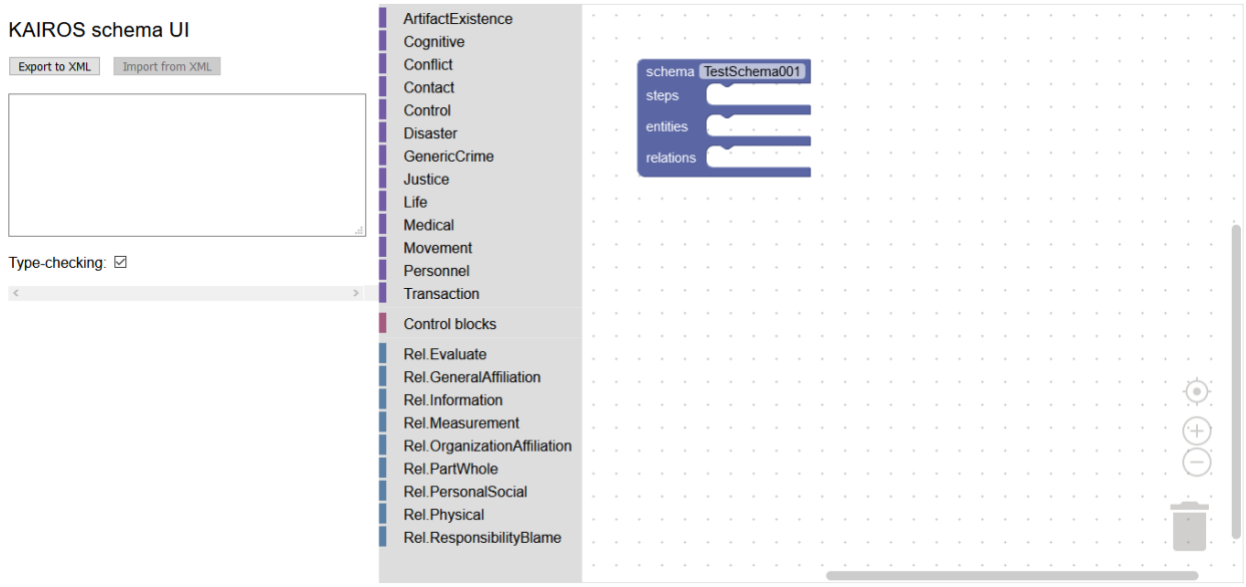
- What **events** occur in this scenario?
    - In the *restaurant* scenario we may have the following events: Entering Restaurant, Getting Seated, Getting Menu, Ordering Food, Eating Food, Paying for Food
- What **entities** appear in the scenario and how do they participate in the events?
    - In the *restaurant* scenario we may have: **Customer** Enters Restaurant, **Customer** is Seated by **Host**, **Customer** Gets Menu from **Waiter**, **Customer** Orders Food from **Waiter**, …

In this annotation project, we will be hand annotating this sort of information about a (pre-selected) set of scenarios into structures called **schemas**. To aid in the building of these schemas will make use of our UI tool, **SchemaBlocks**. In the next section we will go over the features of the SchemaBlocks interface, and how to use it for annotation. After we get familiar with how to use SchemaBlocks, we will go into more detail about the annotation task at hand, and how to use SchemaBlocks to perform the annotation, in addition to some suggestions and tips.

The SchemaBlocks Interface

SchemaBlocks is accessible via this link: https://nlp.jhu.edu/demos/sb and is confirmed to work on the most recent versions (as of Oct 17, 2020) of Chrome and Firefox.

When you open the link, you should see a page like this:



The blue block on the dotted grid is the **schema block**, which is initially empty. You will only have **one** schema block during an annotation session (you cannot create or delete schema blocks). The schema block will contain the entirety of the schema you define. The schema block has three sections:

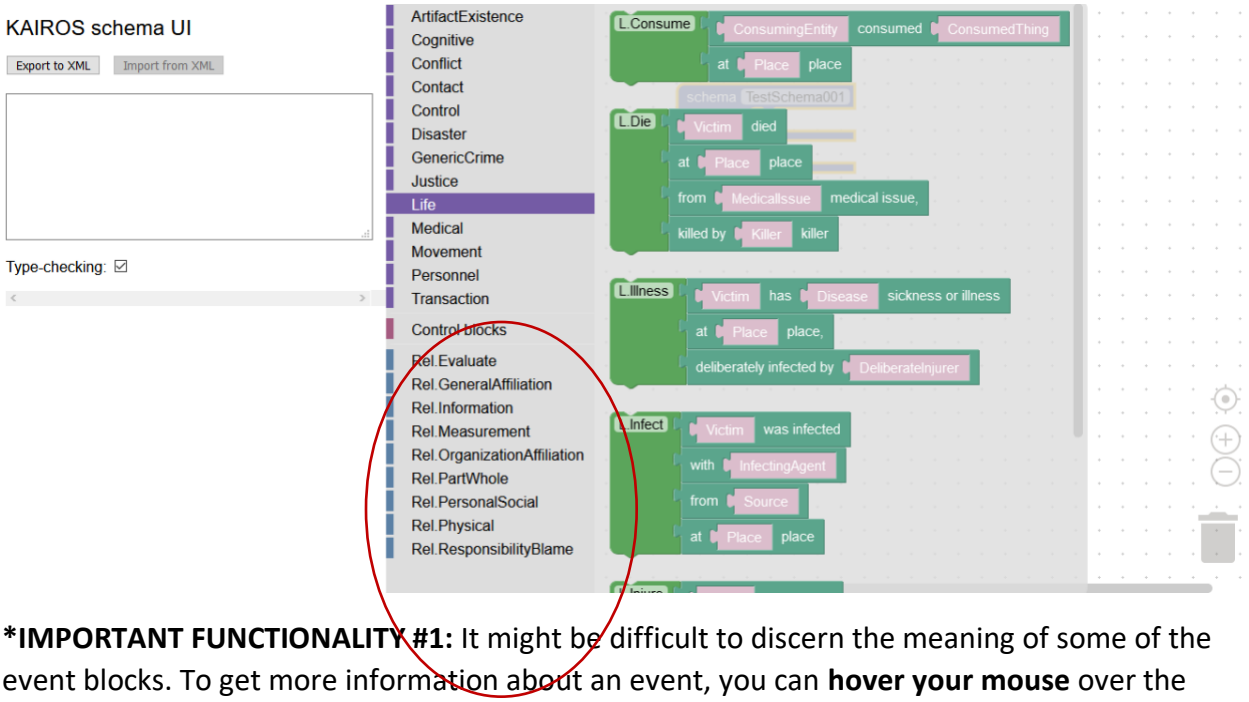
- The **steps** section: In this section, you will drag and drop different **event blocks** which indicate the events that occur in the scenario the schema is about. This will be the most important section.
    - As we will see in the next section, the event blocks will also be used to introduce the entities that appear in the schema.
- The **entities** section: Here you will specify types for entities introduced in the steps section. Note again, you will **not** be introducing entities here! Unlike the other sections, you cannot drag and drop blocks here
- The **relations** section: Here you will drag and drop different **relation blocks** to define pre existing relations that might be true about the entities introduced in the steps section.

We will now go over how to fill out each of these sections in detail.

The Steps Section

In the steps section we will define the **sequence of events** that occur in the scenario we are annotating. We will do this by dragging and dropping **event blocks** into the steps section. SchemaBlocks has a predefined set of 67 different event blocks, each block representing a different kind of event. These event blocks can be thought of as different kinds of 'lego pieces' that we will be stacking together in order to define the events that occur in the scenario. These different blocks are organized under a set of 13 different event categories (circled in red below). Clicking on the event category will give you a list of event blocks belonging to that

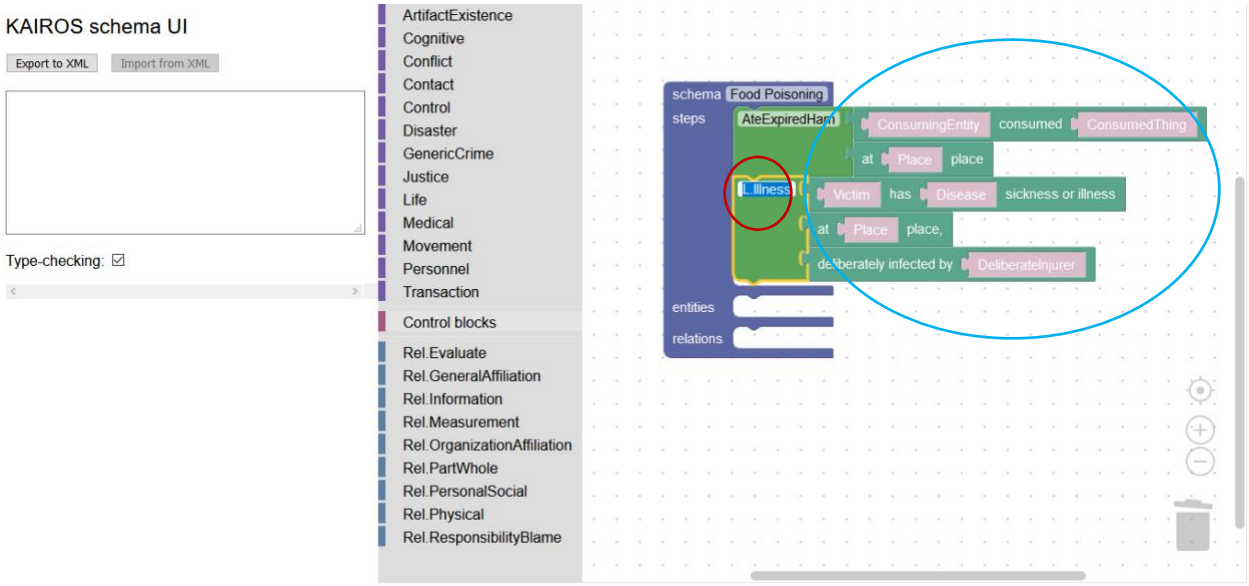
category (note that there is no overlap between categories!). To put an event block in your schema, drag and drop it from the list into the **steps** section.



***IMPORTANT FUNCTIONALITY #1:** It might be difficult to discern the meaning of some of the event blocks. To get more information about an event, you can **hover your mouse** over the darker green section of the event block (this can be done for blocks you have already placed as well).

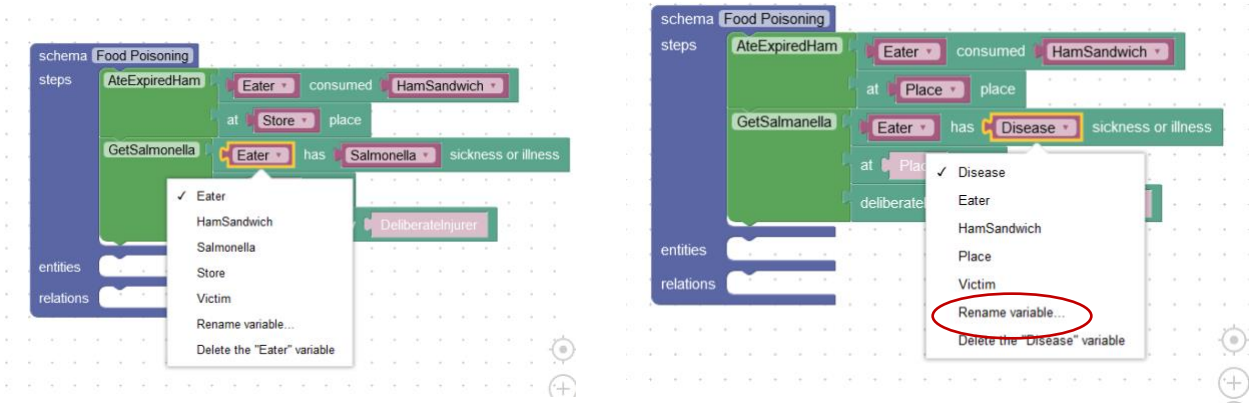
If you have other event blocks in the steps section, the new block will be connected to them automatically. Note that the **ordering of events in the steps section is important**! All event blocks *above* a certain event block are assumed to have happened *before* that event. All blocks *below* a certain event block are assumed to have happened *after*. We will go over more advanced orderings in a future section.

***IMPORTANT FUNCTIONALITY #2:** In both event and schema blocks, you can rename the event or schema by clicking the text box will be the default title of the event or schema (circled in red below), and entering a new title/event name. This will **not** change the underlying type of the event! It will just make it more human readable. Giving human readable names to the schema + all events will be **required for the annotation task**.



All event blocks will have a set of **slots** (circled in blue above) which can be filled with entities that participate in the schema. By default, all slots will initially be **unfilled,** indicated by the greying out of the slot. If the slot is unfilled, it means we are not keeping track of the entity that fills this slot (we will go over when you should fill a slot in the section discussing the annotation task). To fill a slot simply click it. You should notice it "light up." When you fill a slot, you can fill it with either:

- **A pre-existing entity:** You can choose to fill the slot with the name of an entity which already filled a previous slot. This indicates that these entities *are the same*. As an example, below on the left pane, we fill the slot for the second event (an illness) with the same 'Eater' entity that appears in the first event. This indicates that the person who ate the expired ham is the same as the person who contracts Salmonella in the next event.
- **A new entity:** You can choose to fill the slot with a new entity that has not appeared before. This will happen by default when you click to fill a slot, and automatically chooses a "default" name for the new entity. Please do **not** use this default name! If you want to introduce a new entity, you should click Rename Variable to give an informative name to the entity. This is done in the example below on the right side.

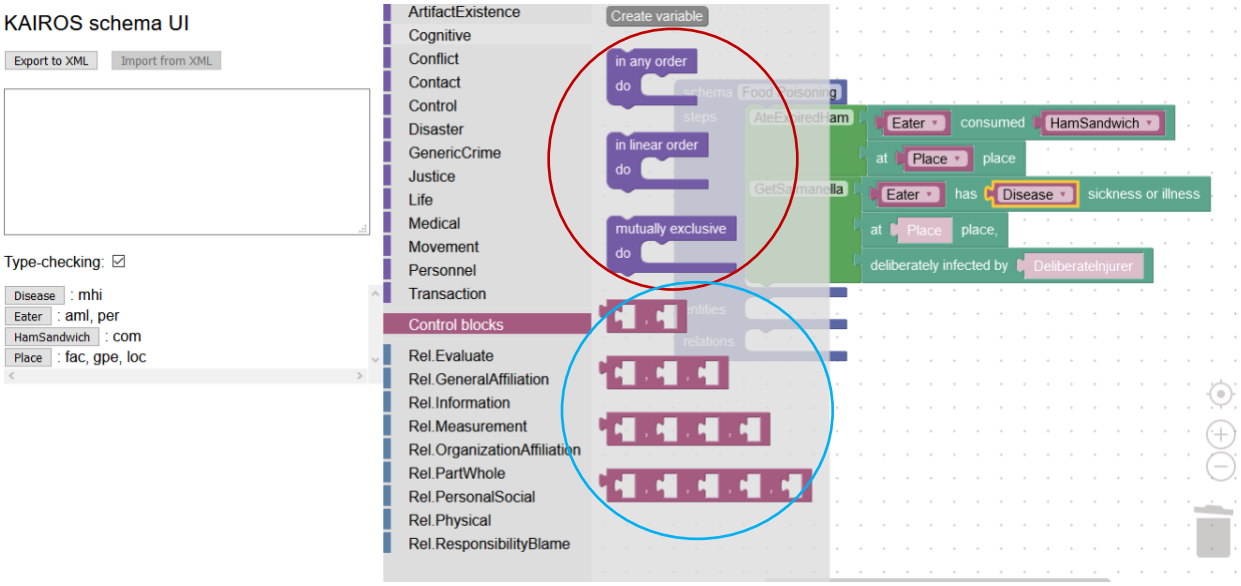


You can “unfill” an accidentally filled slot by clicking “Delete the ___ Variable” in the slot’s drop down menu.

The last piece of info needed for the steps section is understanding the advanced control blocks.

Control Blocks in the Steps Section: Special Orderings, Multiple Entities Filling a slot, etc.

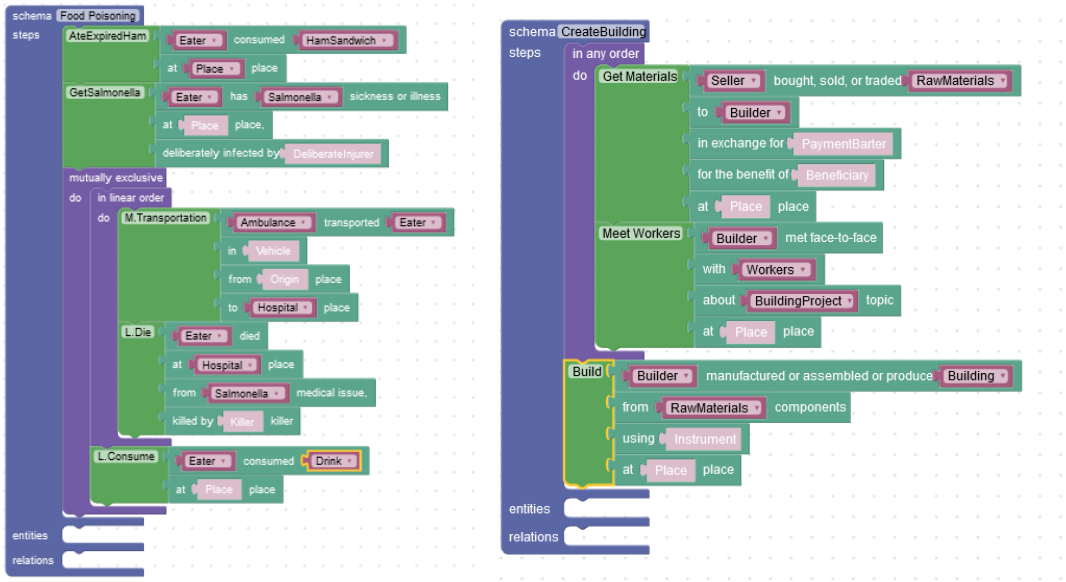
There may be times where a strict linear order of events is not desired in the steps section. For example, you may want to specify that a set of events may occur in any order. There may also be cases where you want to fill a slot with multiple entities. To handle these cases, one can make use of the extra control blocks. There are two types of control blocks: ordering control blocks (circled in red below), and multi-entity blocks (circled in blue below)



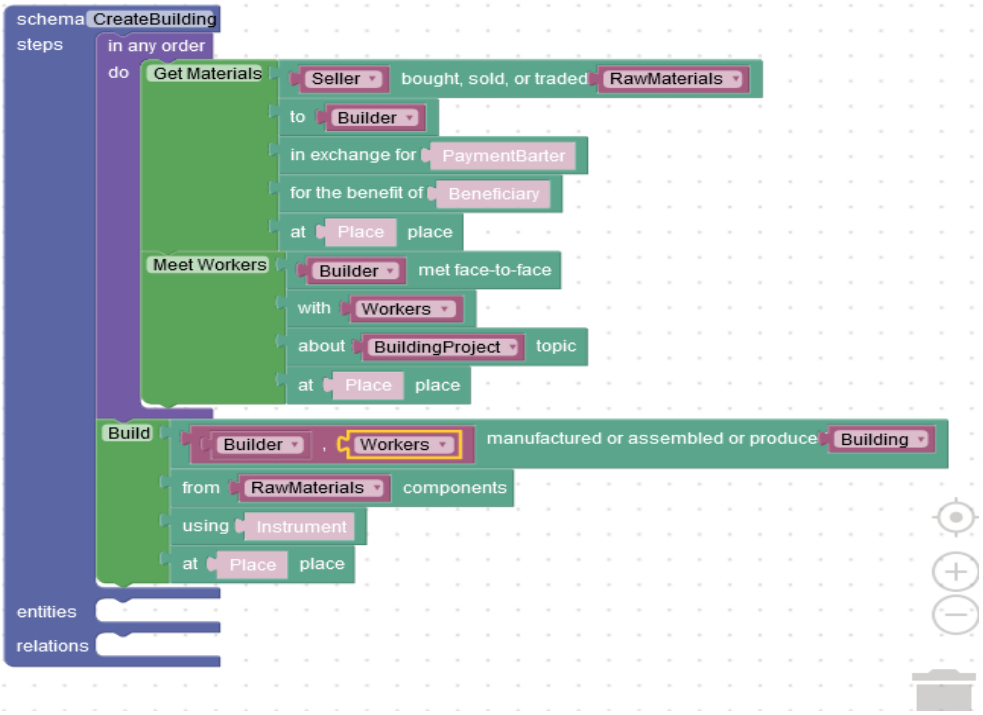
These blocks may be dragged and dropped like any other blocks. First we will go over the different ordering control blocks:

- **In any order** block: This block indicates that all event blocks placed inside may *happen in any order*. All event blocks outside and above/below the in-any-order block are still understood to happen before/after the events in the in-any-order block. An example is given below on the right side. Here the events of Getting Materials and Meeting Workers may happen in any order. The event of Building will occur afterwards.
- **Mutually Exclusive** block: There may be cases where the occurrence of one event precludes the occurrence of another. In these cases we can use the mutually-exclusive block. When events are placed inside of a mutually-exclusive block, we are indicating that only **one** of these events may happen. In cases where separate sequences of events are mutually-exclusive, one can also use the **in-linear-order** block to indicate this. This is best understood by example, given below on the right side. In this unrealistic example, we have marked that one of two things may happen after the GetSalmonella event;

either the person goes to the hospital and dies, or they continue their meal and start consuming their drink. Note that the sequence of going to the hospital and dying are grouped together in the mutually-exclusive block via the in-linear-order block.



We next go over the multi-entity blocks. These can be used in cases where multiple entities may fill a slot. For example, in the CreateBuilding example above, we may want to indicate the both the Builder entity and the Workers entity partake in the building event. We can use the multi-entity blocks as below to mark this:



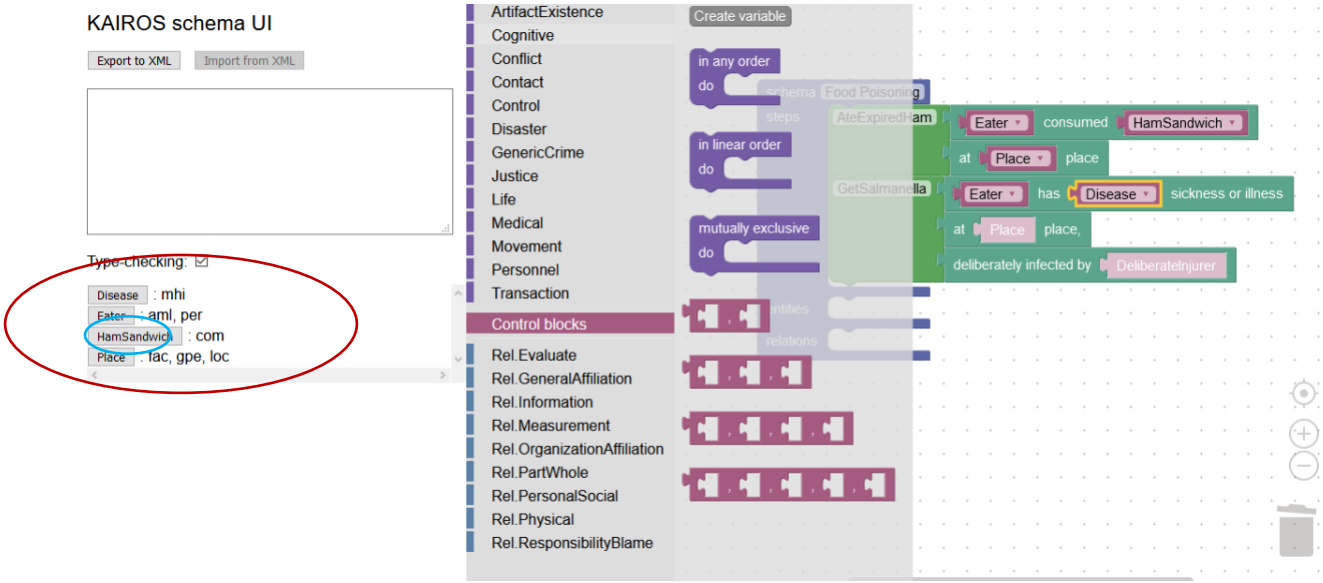
Note that this may require some duplication of the previous entities (for example, duplicating the Worker entity, and placing the duplicate in the extra slot).

***IMPORTANT FUNCTIONALITY #3:** You can copy any block or entity by right clicking it and selecting Duplicate.

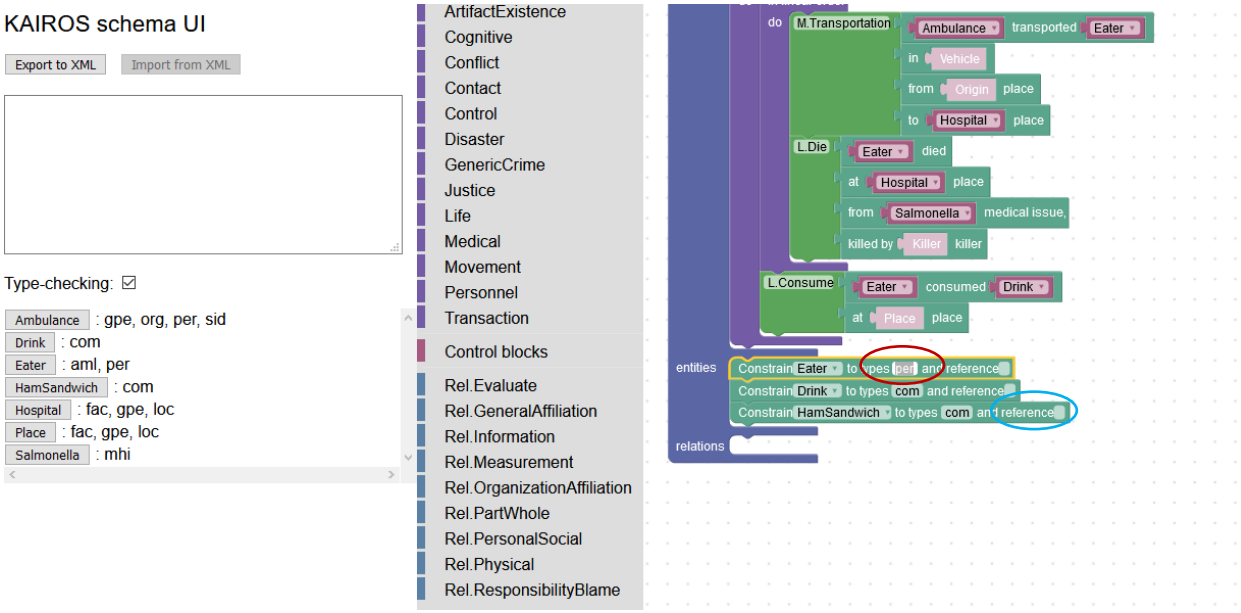
That's about it for the steps section, let's take a look at the **entities** section

Entities Section

Unlike the steps section, there will be no dragging a dropping of components here. This section will be used to place type constraints on the entities in the schema. Entity **types** give information about what the entity is (is it a person? A country? …). By default, entities are given a preset set of types depending on where they were first introduced. This can be seen in the **Typing Pane**, circled in red below



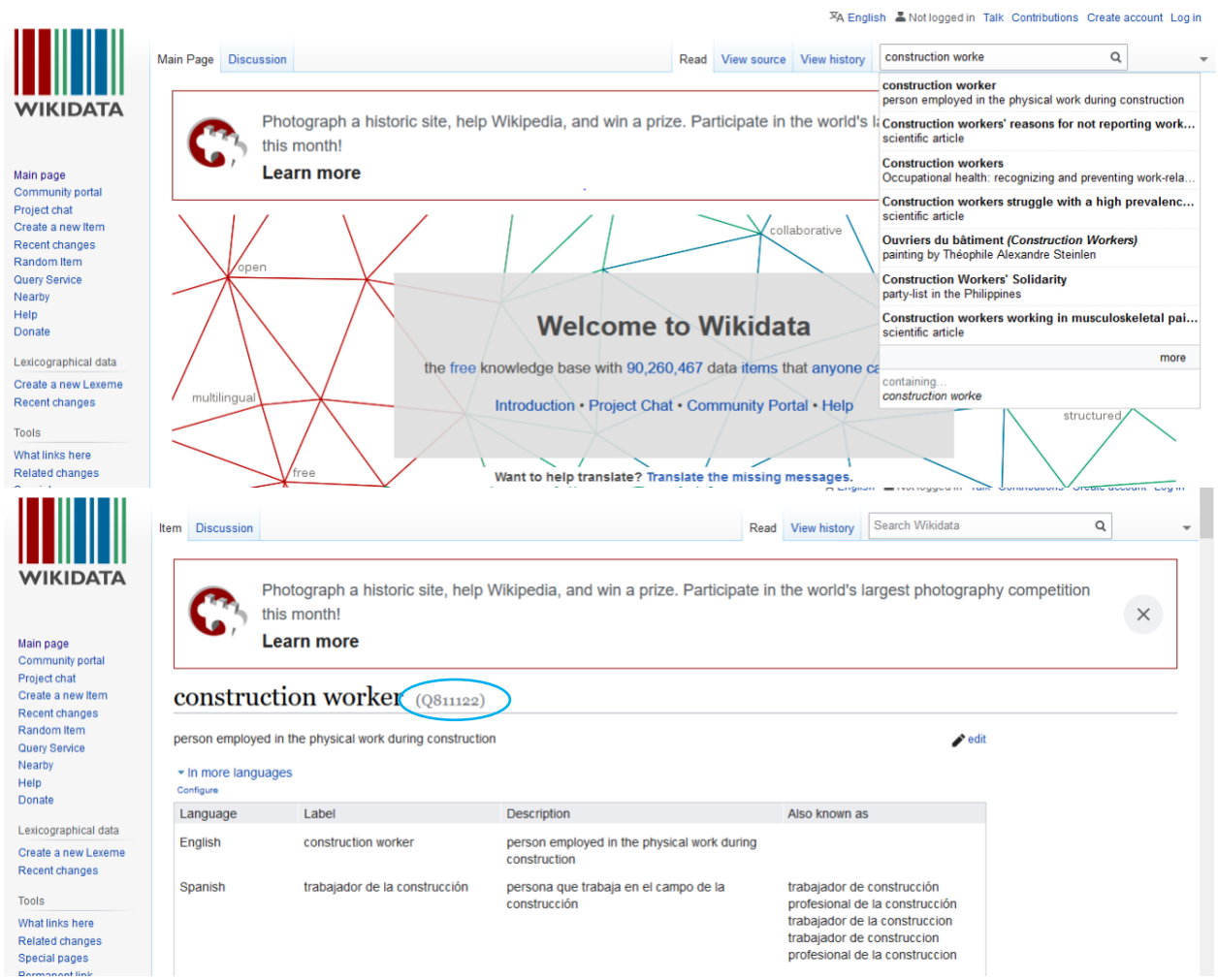
To see what the type abbreviations stand for, hover your mouse directly over the abbreviation in the Typing Pane. It will almost always be the case that you will want to refine these default types. To do this, click on the entity's button (circled in blue above) in the Typing Pane. You should see the entity appear in the Entities section of the schema block. You can restrict the types of the entity by modifying the text in the 'types' section of its corresponding block in the entities section. An example is given below circled in red, where we have modified the types for the Eater entity to just a 'person' entity type (you will almost always be simply paring down the default types, rather than typing new ones in):



The predefined types are oftentimes not fine grained enough to be fully informative. In some cases, this may be very important for the schema definition. For example, in our example Building schema, it may be important to specify that the Worker entities are of the specific type *construction worker*, rather than just the, rather non-informative, type of *person.* We can specify even more specific types by using the **reference** slot in the entity blocks; these are circled in blue in the above example.

The **reference** slot allows us to link to types in the online database Wikidata (accessible from **wikidata.org**). If you suspect that a more specific type will be desirable, you can use wiki data as follows:

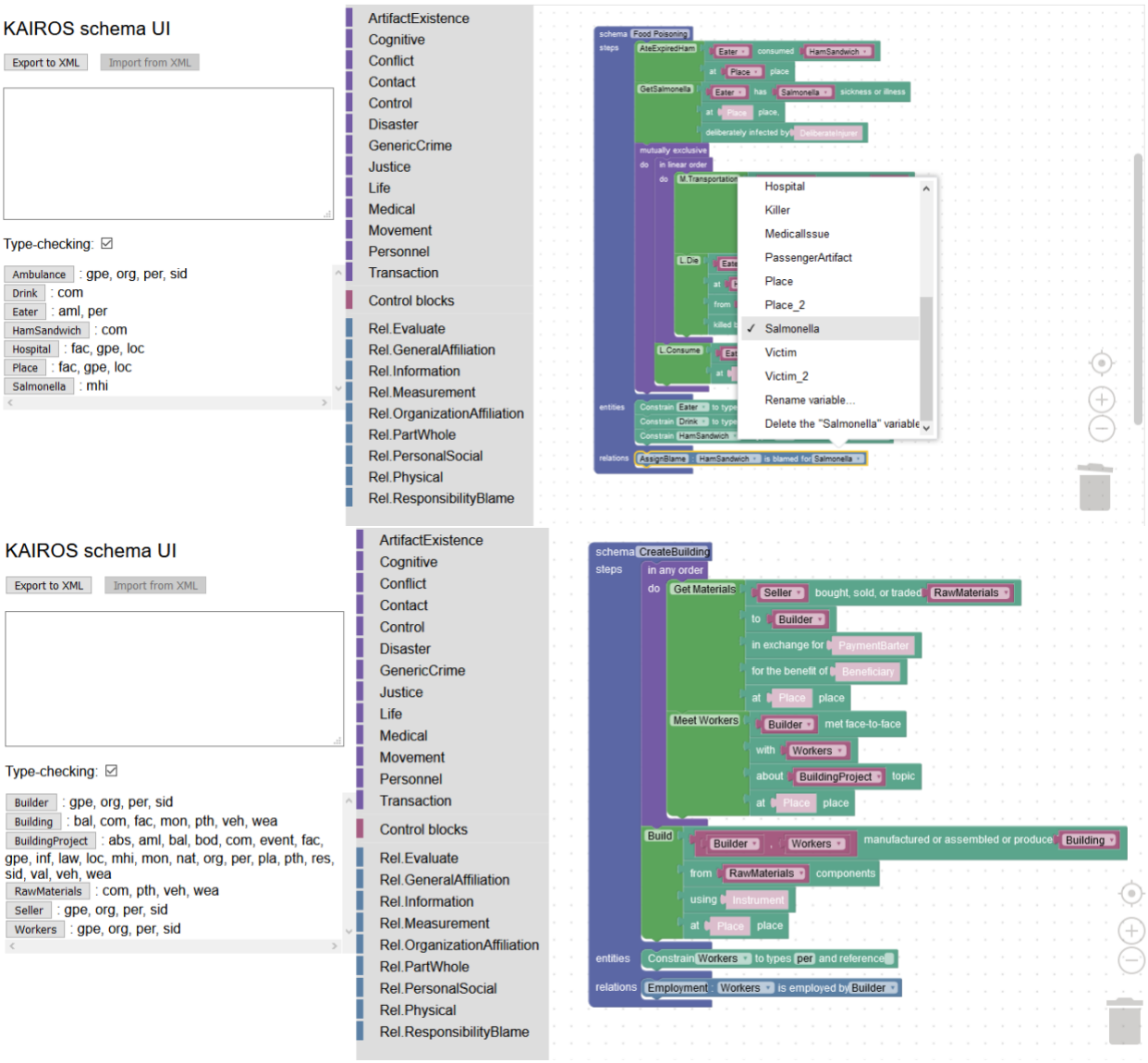
- In the search bar on the wikidata home page, enter in the name of the specific type (in our example, construction worker). If this type exists in wikidata, it should pop up automatically. If it doesn't, try a different term, or perhaps make your type less specific.
- If the type showed up, then click on it to go to its page. Then copy and paste its **id** and put it in the reference slot of your schema. The location of the type **id** on wikidata is circled in blue in the example below.



Relations Section

The relations section is used to indicate relations that may hold between entities that appear in the schema. Similar to the steps section, we will be dragging and dropping blocks into this section. However, rather than dropping event blocks, we will be dropping **relation blocks** into this section. Like the event blocks, the relation blocks are predefined and categorized into several categories. The relation blocks can be dragged and dropped into the relations section in the same way that event blocks are placed in the steps section. One should note that in the relations section, **order does not matter**.

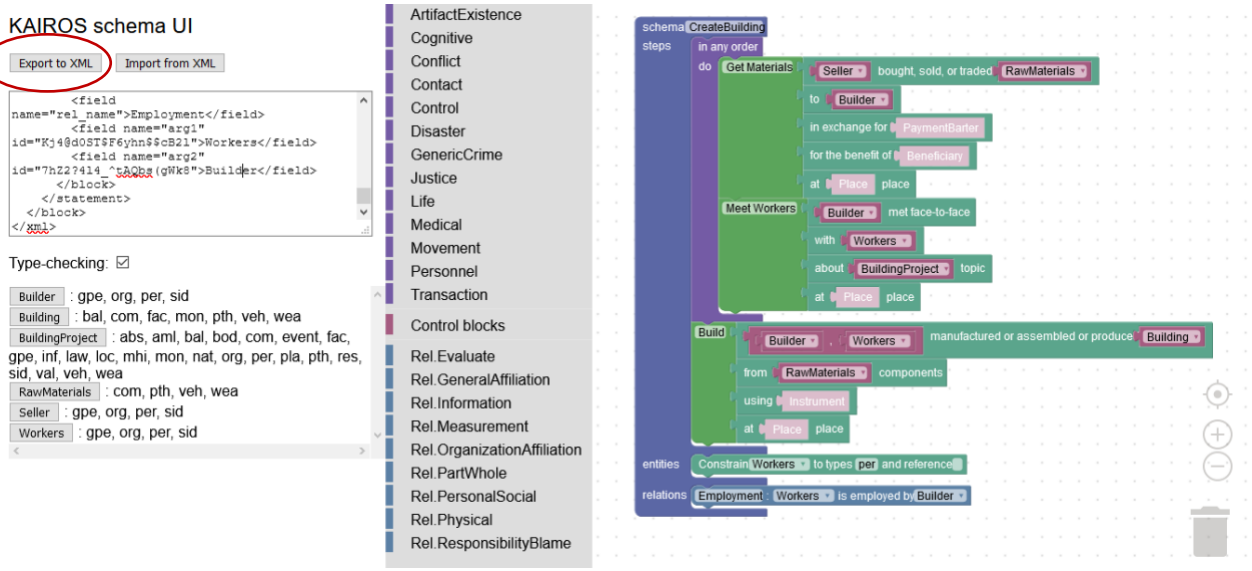
Once the relation block is placed in the relations section, one can use the drop down menus to select pre-existing entities that belong in the relation. An example of this is presented below:



Exporting Your Created Schema

After you have finished your schema it is time to export your hard work into a machine readable format. To do this, click the 'Export to XML' button (circled in red below). Upon clicking this button, you will see text appear in the box below the button. Copy and paste this text to a normal text file. This will be the output to hand in.

If you want to continue to work on a schema that you had saved in a XML file, you can copy and paste its XML markup in the same text box and then click the 'Import to XML' button. This will load up the graph from the pasted XML.


KAIROS schema UI
Export to XML
Import from XML
<field name="rel_name">Employment</field>
<field name="arg1" id="Kj4@d0ST$F6yhn$$cB2l">Workers</field>
<field name="arg2" id="7hZ2?4l4_^tAQbs(gWk8">Builder</field>
</block>
</statement>
</block>
</xml>
Type-checking:
Builder : gpe, org, per, sid
Building : bal, com, fac, mon, pth, veh, wea
BuildingProject : abs, aml, bal, bod, com, event, fac, gpe, inf, law, loc, mhi, mon, nat, org, per, pla, pth, res, sid, val, veh, wea
RawMaterials : com, pth, veh, wea
Seller : gpe, org, per, sid
Workers : gpe, org, per, sid
ArtifactExistence
Cognitive
Conflict
Contact
Control
Disaster
GenericCrime
Justice
Life
Medical
Movement
Personnel
Transaction
Control blocks
Rel.Evaluate
Rel.GeneralAffiliation
Rel.Information
Rel.Measurement
Rel.OrganizationAffiliation
Rel.PartWhole
Rel.PersonalSocial
Rel.Physical
Rel.ResponsibilityBlame
schema CreateBuilding
steps
in any order
do
Get Materials
Seller
bought, sold, or traded
RawMaterials
to
Builder
in exchange for
PaymentBarter
for the benefit of
Beneficiary
at
Place
place
Meet Workers
Builder
met face-to-face
with
Workers
about
BuildingProject
topic
at
Place
place
Build
Builder
Workers
manufactured or assembled or produce
Building
from
RawMaterials
components
using
Instrument
at
Place
place
entities
Constrain
Workers
to types
per
and reference
relations
Employment
Workers
is employed by
Builder