# Documentation Experiment 2b and Task Environment

## How to run the code

Have the following installed/on your computer:

* TypeDB
* TypeDB Studio
* The ontology schema (CP\_ontology\_schema\_corrected.tql) and basic instances (Basic\_Instances.tql) files
* A Python IDE
* The code from the ontology\_development branch at GitHub: [GitHub - xDaya/Co-Learing\_Scenario at ontology\_development](https://github.com/xDaya/Co-Learing_Scenario/tree/ontology_development)

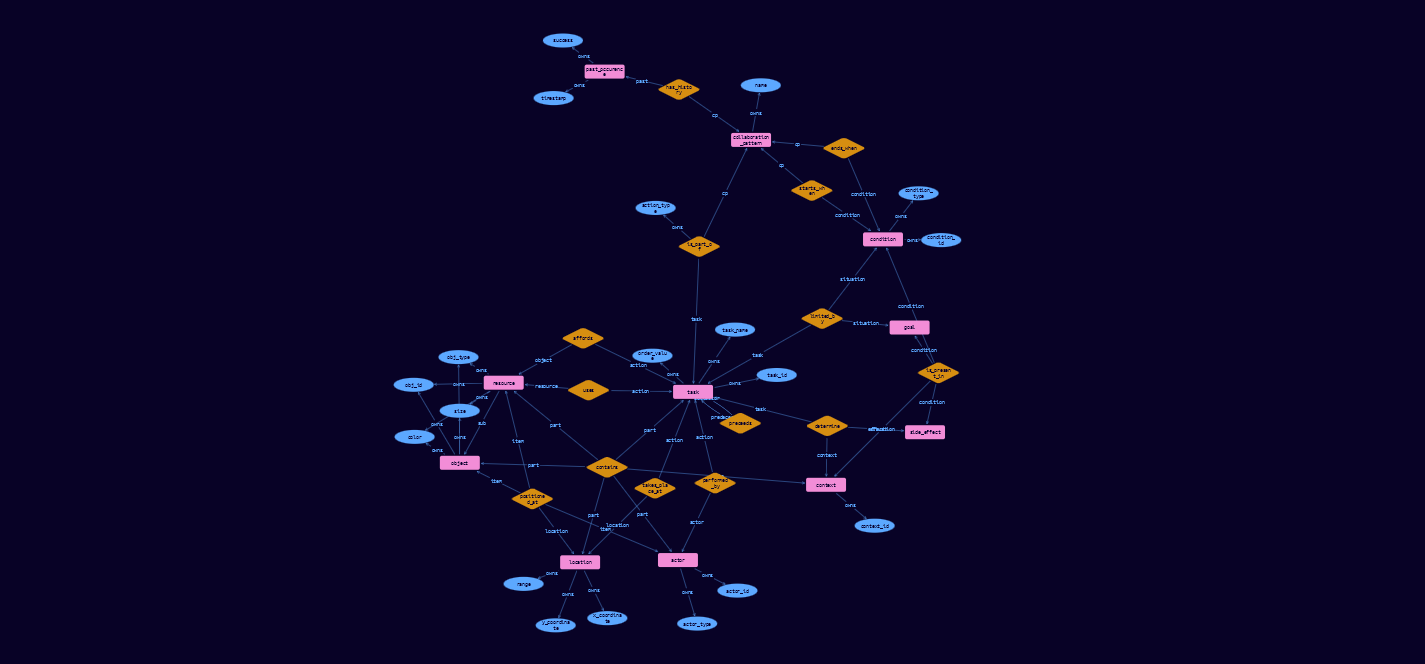
### Step 1. Make sure the ontology is up and running

Run an instance of the TypeDB server (for documentation, see [Install and Run TypeDB | Vaticle](https://docs.vaticle.com/docs/running-typedb/install-and-run))

Open TypeDB Studio and create a database that is called *CP\_ontology*.

Write the schema to this database (CP\_ontology\_schema\_corrected.tql), then write the basic instances (Basic\_Instances.tql) as data to this database. For more documentation on TypeDB Studio, see [Quickstart | Vaticle](https://docs.vaticle.com/docs/studio/quickstart).

To check whether this was done correctly, you can run the query *match $x sub thing;* to get a visualization like the one below:



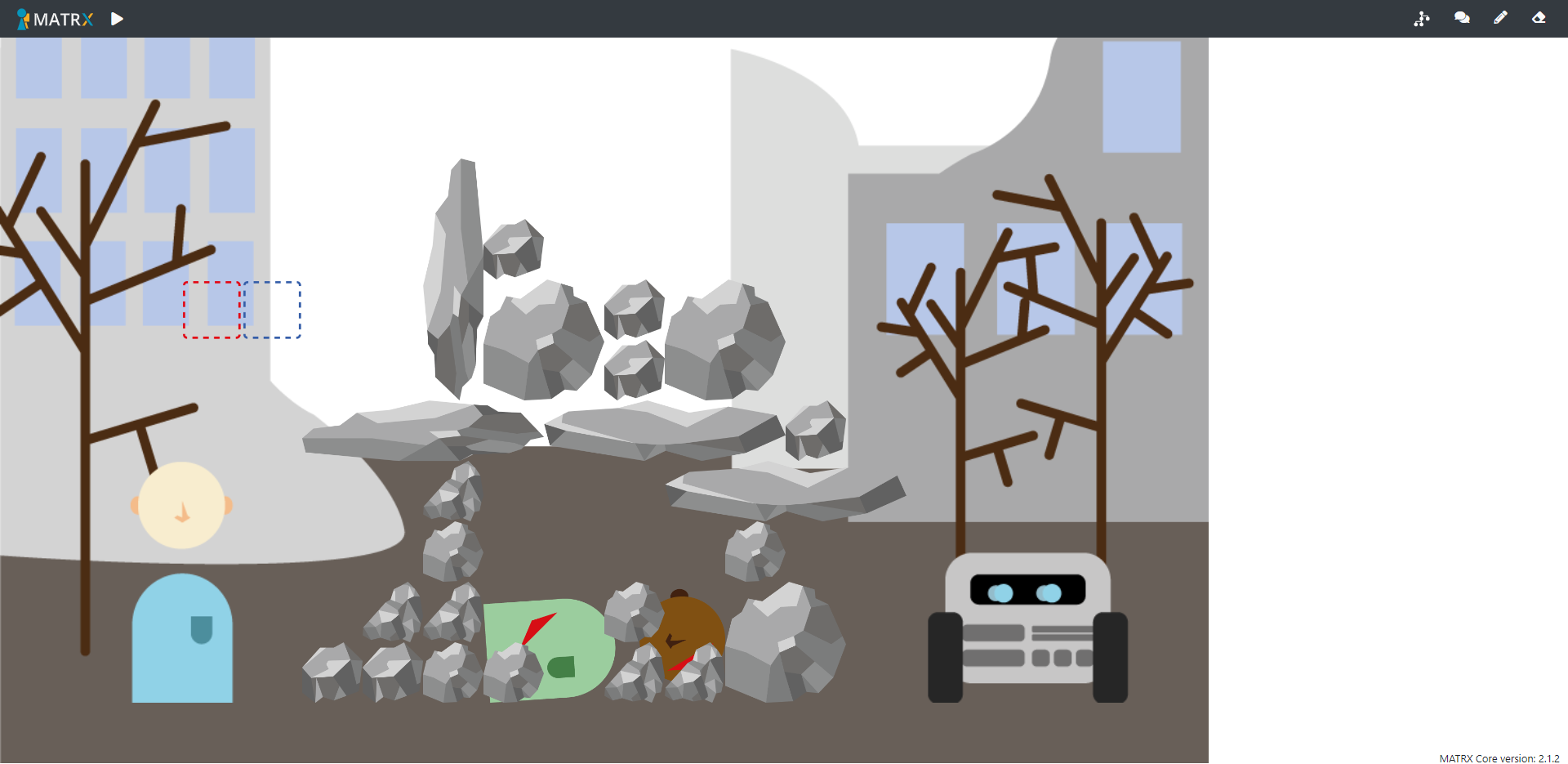
### Step 2. Check the Python code

Before running the Python code from your IDE, check the URL to the images of the task environment (*main.py*, line 22). Make sure it refers to the main project folder (*/Co-Learing\_Scenario*).

### Step 3. Running the task environment

Now you can try running the Python code by running *main.py*! If the TypeDB server is not running in the background, you will get an error message.

To view the task, open your browser and go to [*http://localhost:3000/human-agent/human\_selector*](http://localhost:3000/human-agent/human_selector). It should look like this:



You can open the GUI for interacting with the ontology by clicking the graph icon in the top right corner.

## Experiment procedure outline

Every participant will go through a total of 8 rounds of playing the task. Every round will have a different scenario, but the scenarios are grouped in two types of scenarios; one in which breaking rocks can have severe negative effects, and one in which there is a brown rock that cannot be picked up. Before starting the experiment, participants will have the opportunity to practice the task in a simple scenario without the robot.

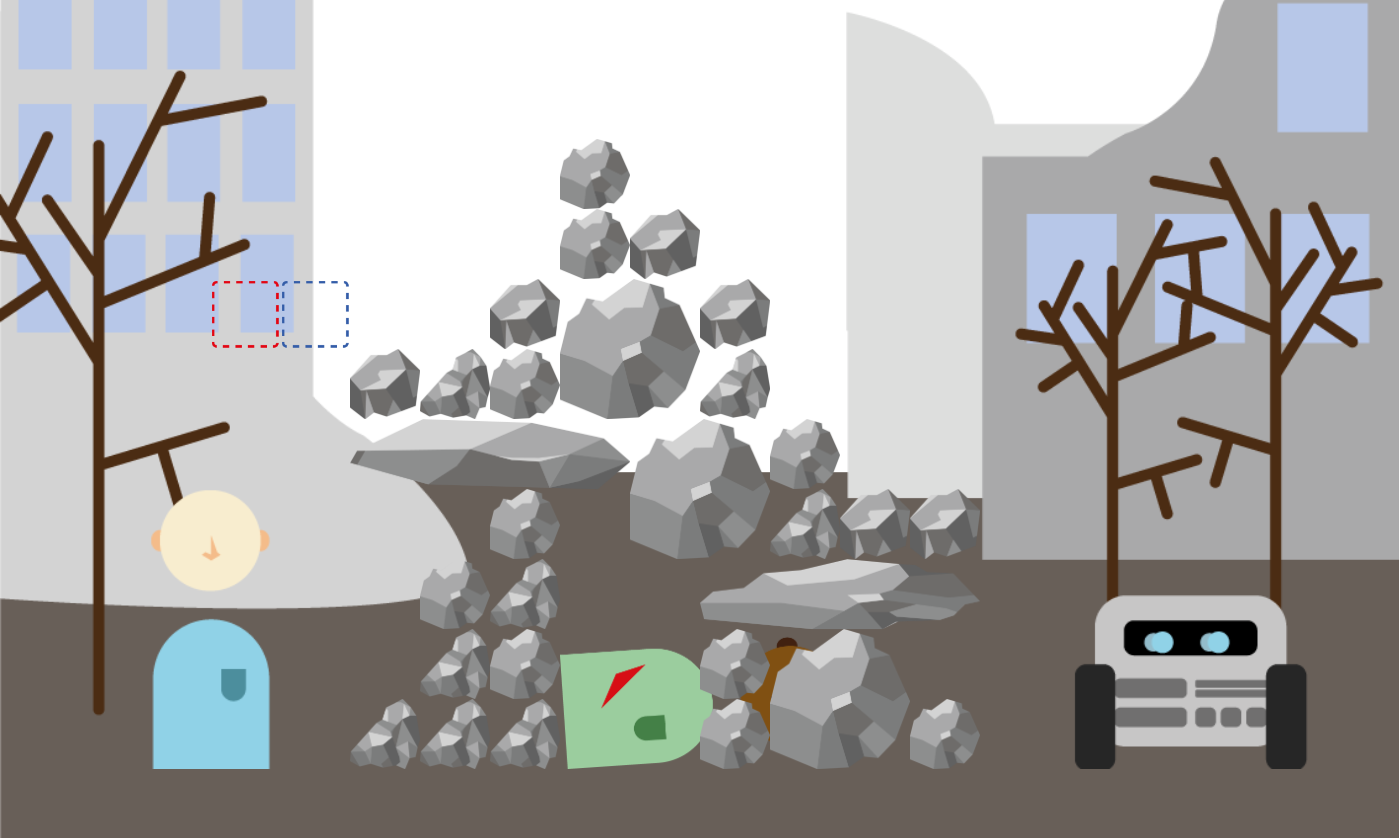
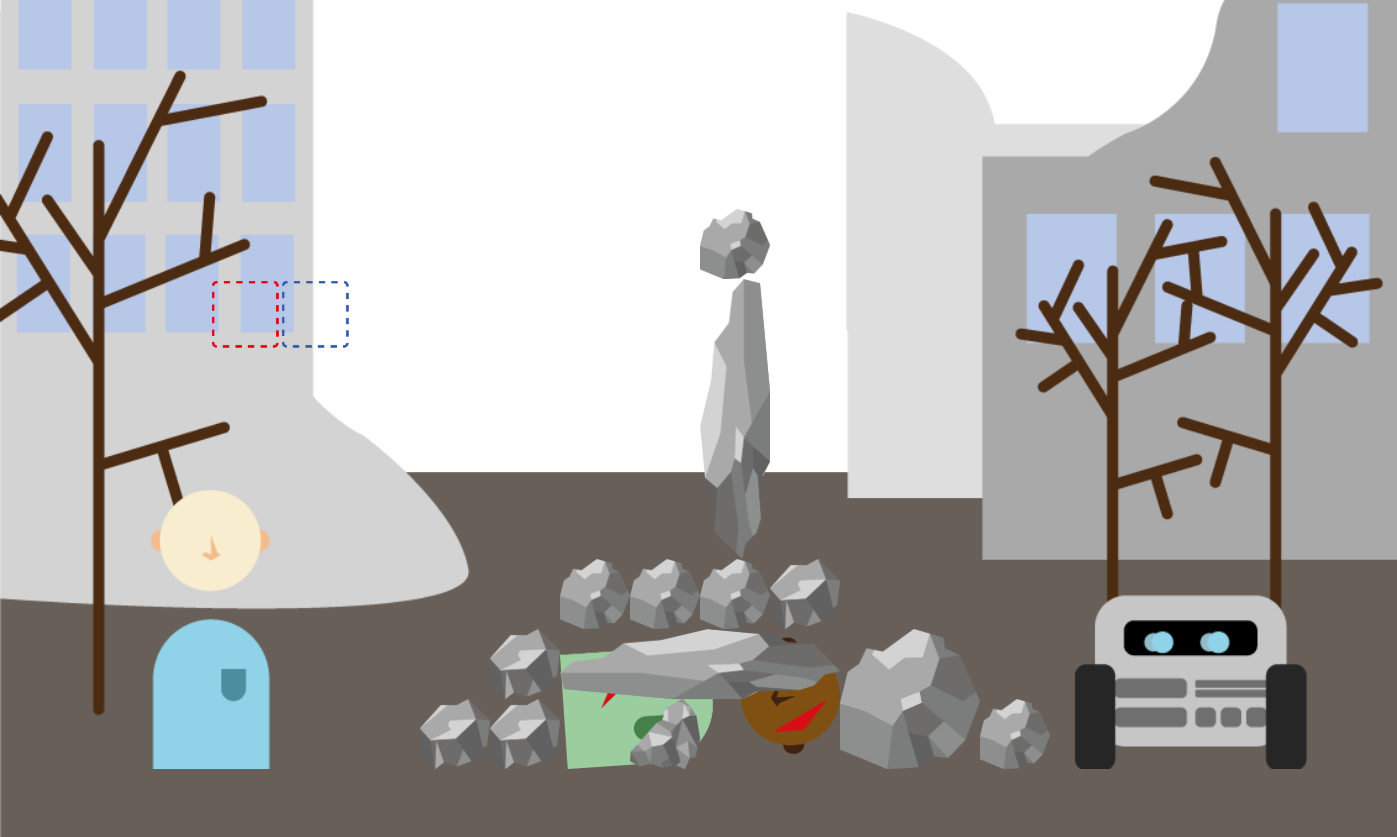
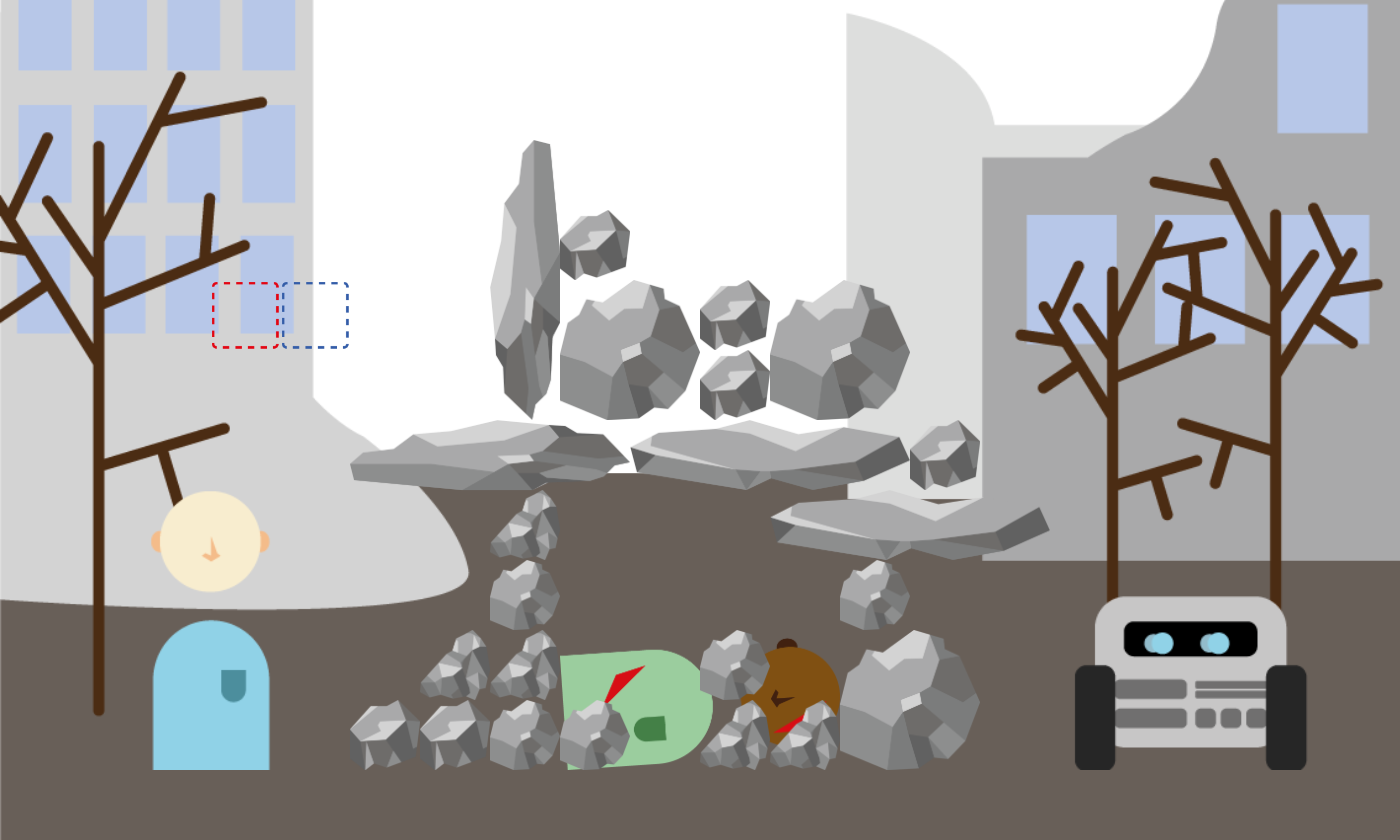
For each scenario, the participant will go through the following steps:

* Perform the task (i.e. try to save the victim as quickly as possible). During this, they can pause the task to store Collaboration Patterns.
* Receive a prompt to describe relevant Collaboration Patterns if they want to.
* Answer a few short questions (details to be determined).

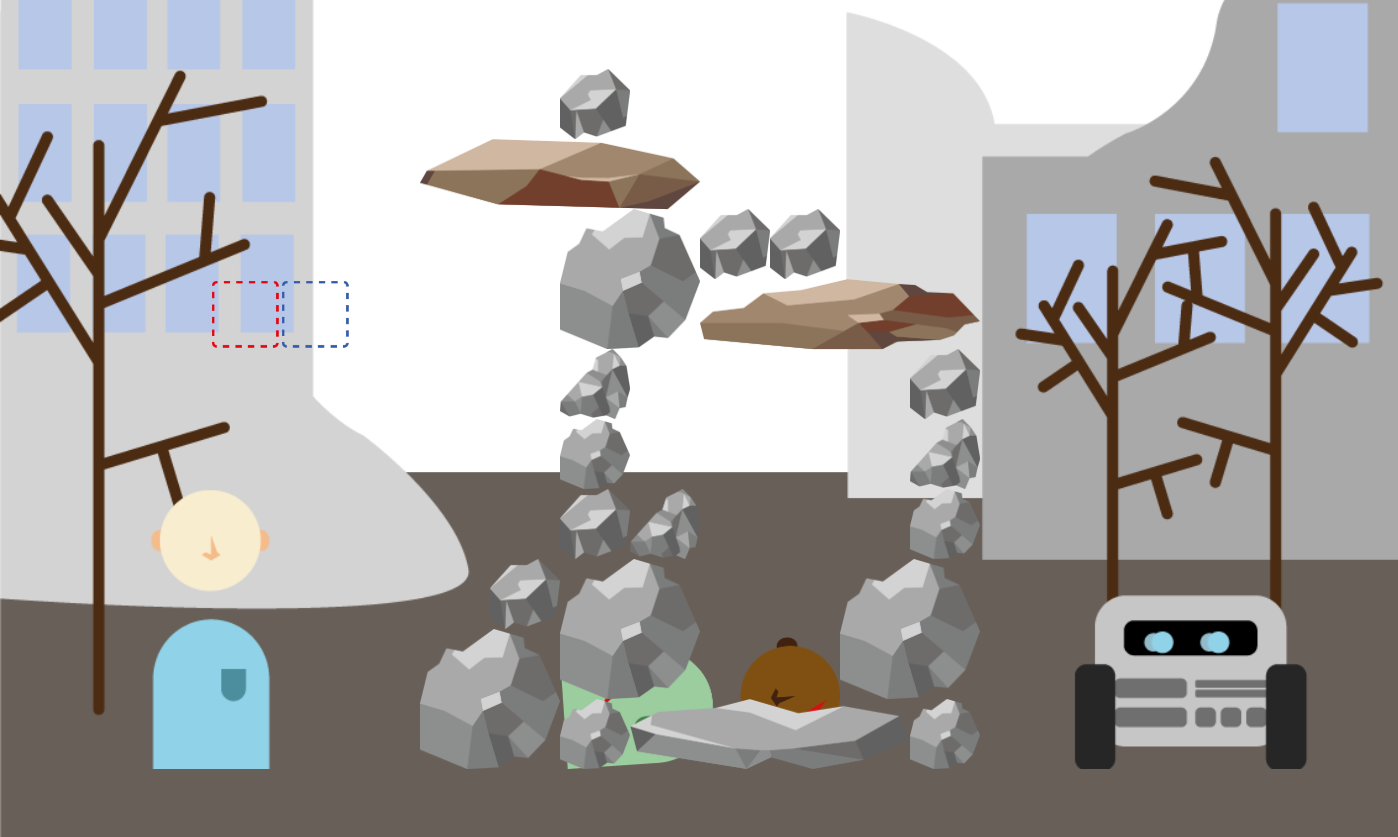
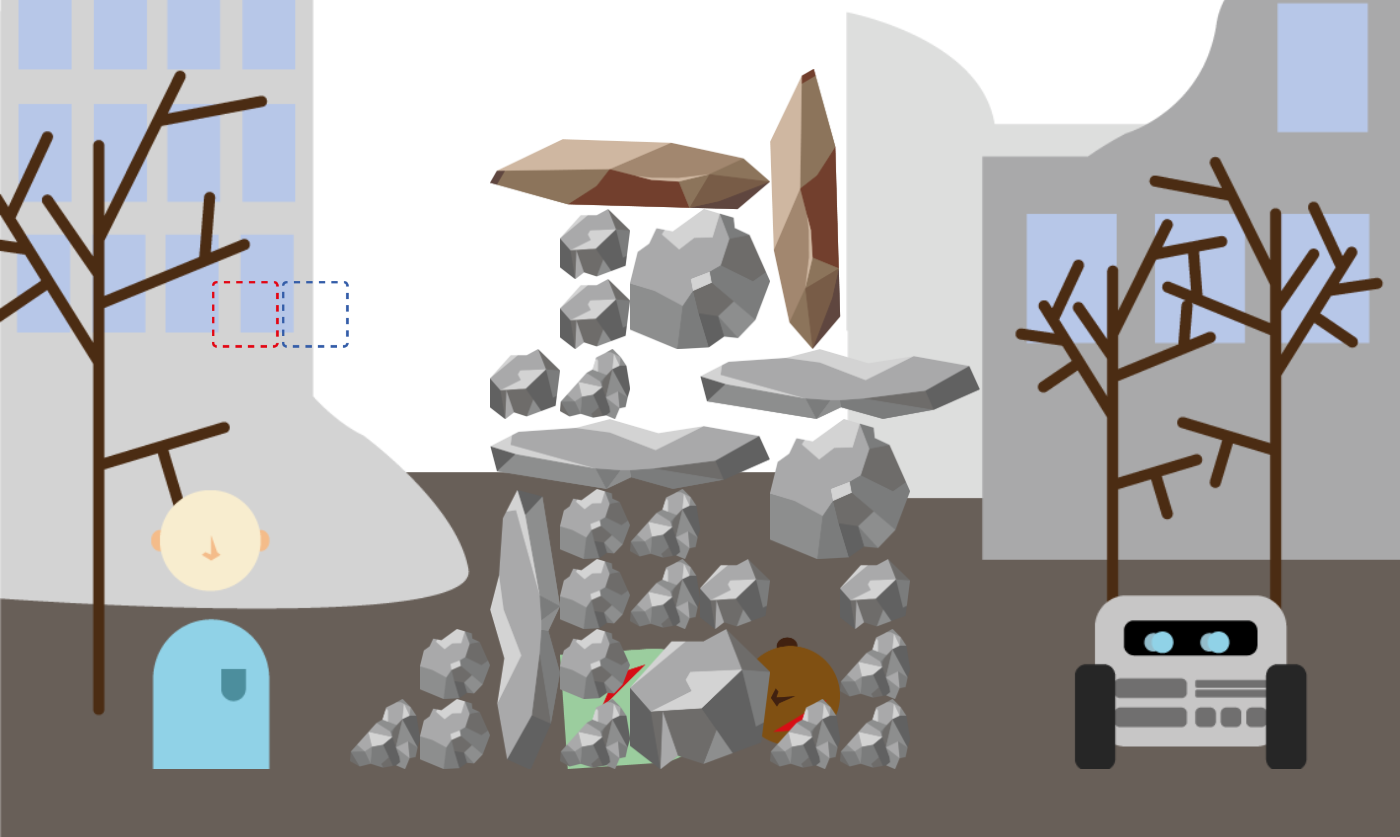
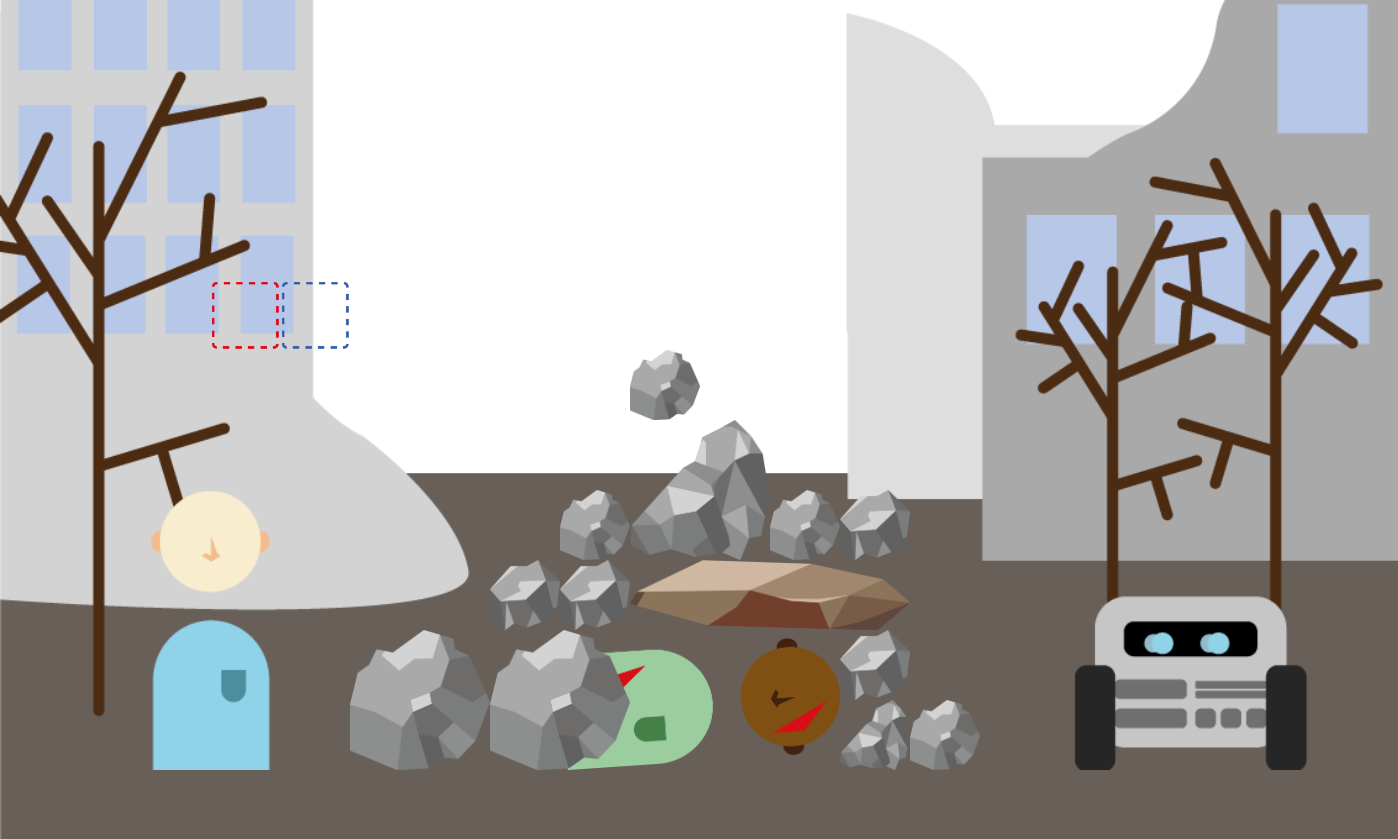
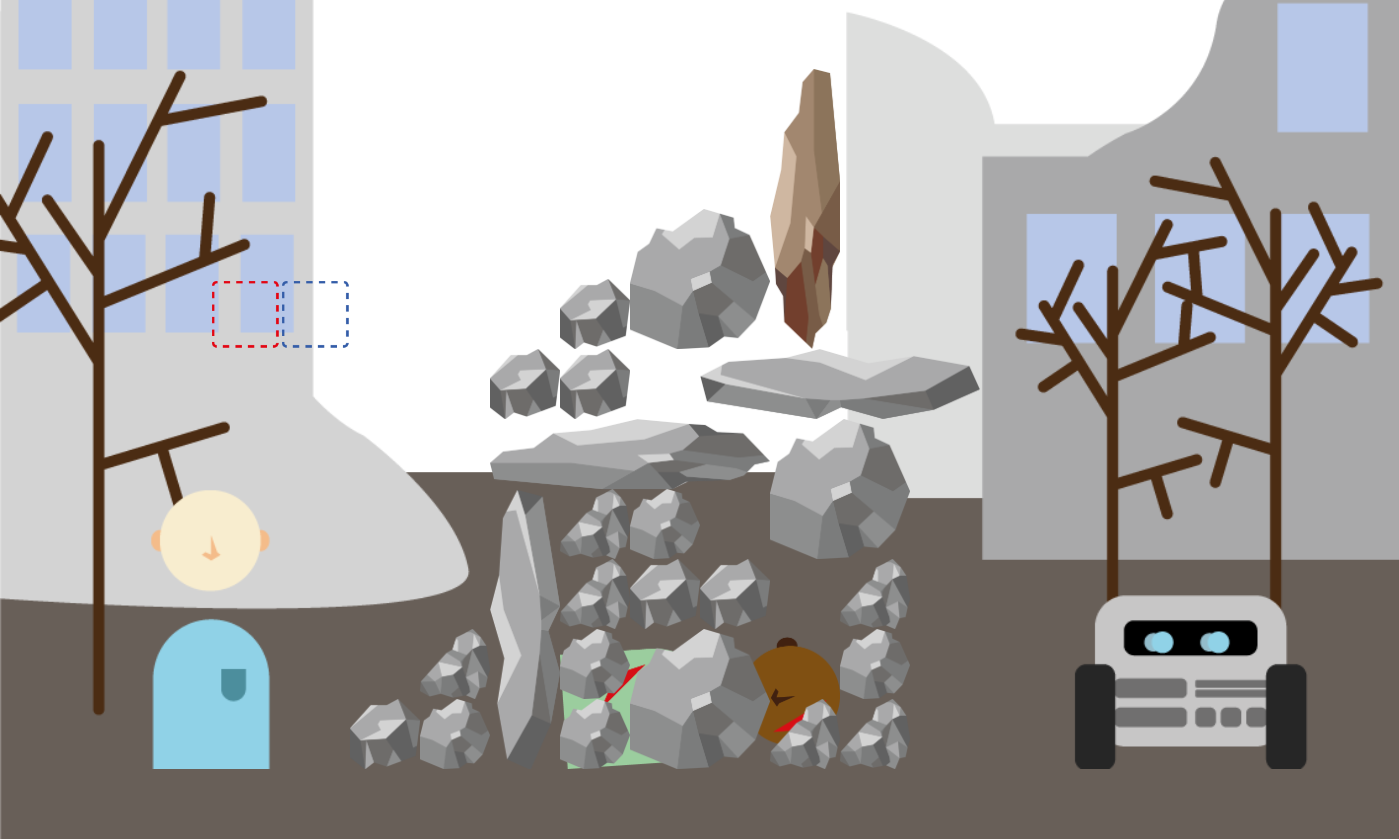
After the 8 rounds, participants will be asked to answer some more questions.

### Experiment scenarios

Don’t break scenarios:



Brown rock scenarios:



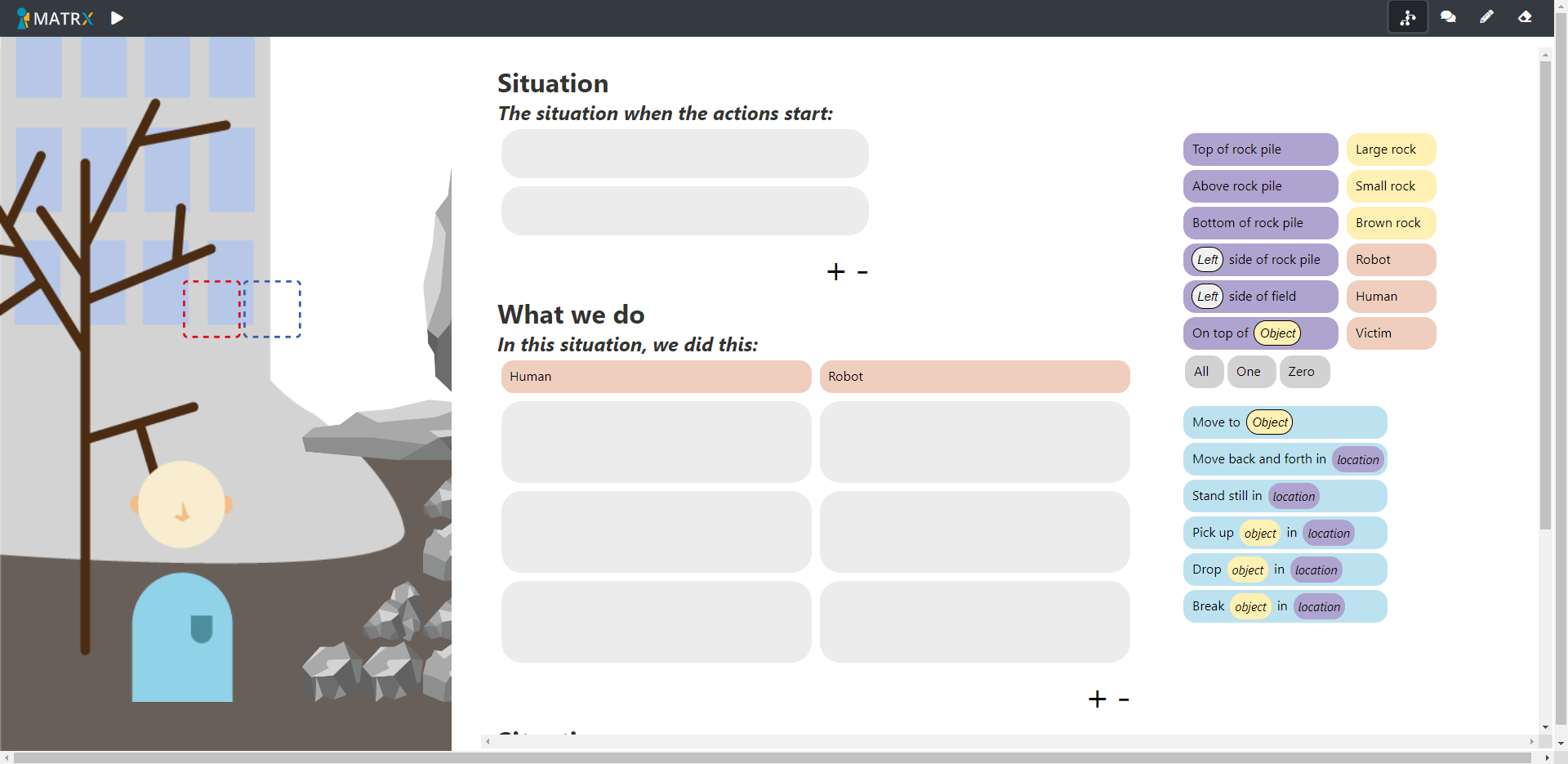
### Experimental Conditions

|  |  |  |
| --- | --- | --- |
| **Baseline** | **Basic Communication** | **Ontology + GUI** |
| Based on the original experiment (but with the new agent behavior)  No direct communication  Learning about the other happens by behavioral observation only | The baseline with additional communication about actions  Both human and robot communication are automated  ‘I am picking up a large rock’  Message sent when action is executed. | The baseline with use of the ontology and GUI  Human can describe and therefore communicate Collaboration Patterns, that can be used by the robot.  The communication from the ‘basis communication’ condition is also present.  The robot communicates which CP they will execute. |

## Interaction flow

To store patterns in the ontology, participants can use the GUI in the below images. There are a few important aspects to consider:

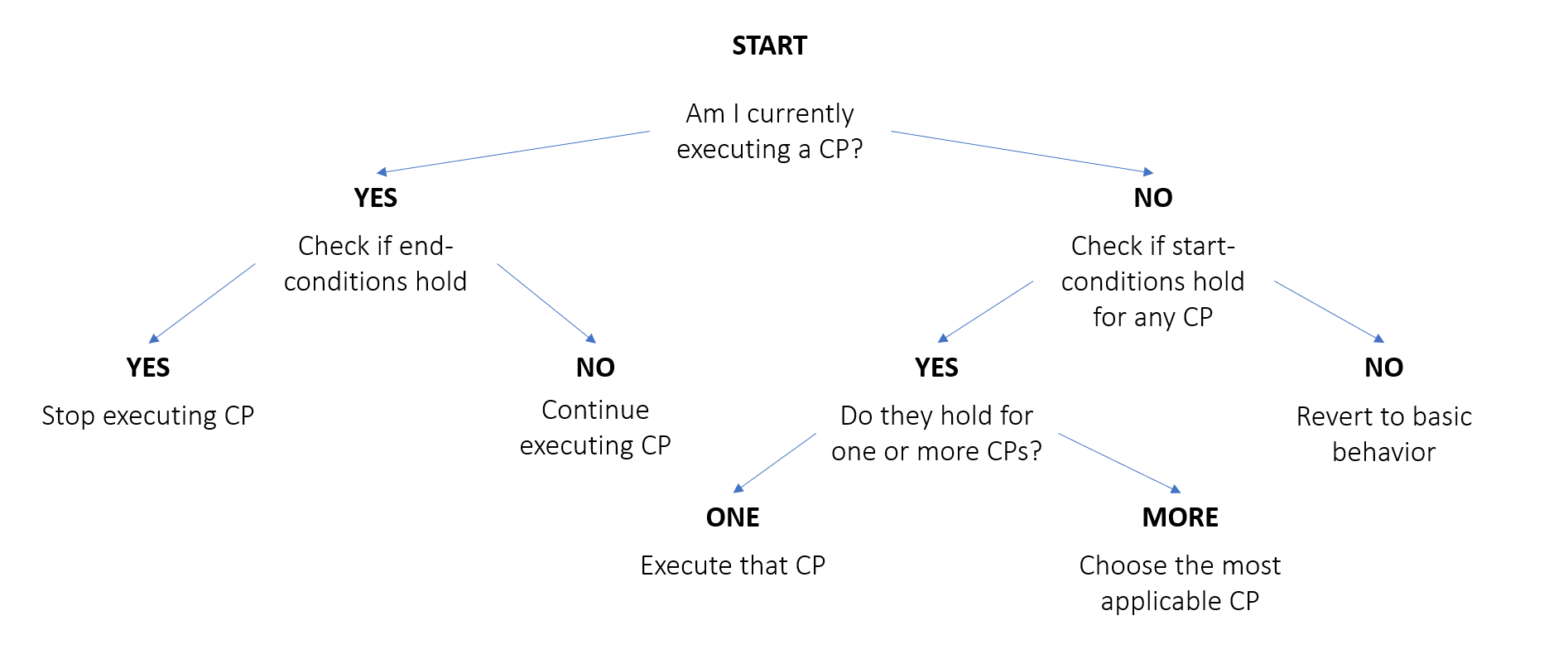
* When the GUI is opened, the task is automatically paused. To resume the task, the play button must be pressed.
* To send a newly described Collaboration Pattern to the ontology, two steps are necessary:
  + Press ‘Send’
  + Resume the task
* Once the Collaboration Pattern is stored, the robot will be able to access it and execute it.





## Robot behavior tree

The robot uses a behavior tree to decide how to behave. This behavior tree refers to parts of behavior that are rule-based, as well as parts of behavior in which decisions are made probabilistically, based on rewards. The high level behavior tree looks roughly as follows:



In this behavior tree, the leaf nodes ‘Choose the most applicable CP’ and ‘Revert to basic behavior’ contain lower level decisions made based on learning via rewards.

## Learning mechanism

The learning mechanism implemented in the parts mentioned has not thoroughly been tested, and the reward update mechanism does not yet include the correct formulas. However, the basic mechanism of storing a value for state-action pairs and doing reward updates after performing an action has been implemented. Currently, actions are chosen in a greedy manner. When testing, you will notice that will cause some problems here and there that need to be resolved later on.

### State representation

The state representation currently used is one that represents all high-level locations as used in the GUI (e.g. Top of rock pile), and whether there is any small rock, large rock and/or brown rock present in that location. When such an object is present, it is represented by a 1, when it isn’t, it is represented by a 0. This means that the state representation consists of a series of 18 binary digits, as follows:

[[1, 1, 1], [1, 1, 1], [1, 1, 1], [1, 1, 1], [1, 1, 1], [1, 1, 1]]

### Choosing the most applicable CP

Every time the start conditions of several Collaboration Patterns hold at the same time, the robot will have to choose which one to use. It will check whether there are any expected reward values stored in the current state for these CPs, or whether they are stored in a near state.

After choosing the CP with the highest expected reward, that CP will be executed in a rule based manner until the end conditions hold. A reward update is performed after every run through the CP, as well as when the end conditions hold. Breaking out of a CP that you’re stuck in can be achieved by defining a minimum expected reward; this has not yet been implemented though.

The reward buildup is as follows:

* The decrease in distance to the goal state, discounted by:
  + Victim harm
  + Idle time

The learning mechanism is supposed to be based on a contextual bandit formulation.

### Basic behavior

When the start conditions hold for none of the Collaboration Patterns, the agent will perform some basic behavior. This basic behavior is built up of the following actions:

* Pick up
* Drop
* Break
* Stand still
* Move back and forth

For the first three actions, choosing an object and moving towards it is included in the actions in a rule-based manner. Reward updates are done after every action execution.

The reward buildup is as follows:

* A basic reward based on whether there was a state change or not, discounted by:
  + Victim harm

The learning mechanism is supposed to be based on a Q-learning formulation.