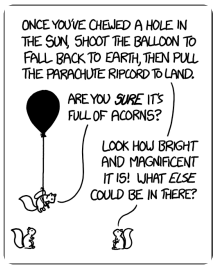


(https://xkcd.com/1503/)
Squirrel Plan

This assignment is due on Tuesday, March 1, 2022 at 11:59PM EST.

You can download the materials for this assignment here:

- Example of How To Convert WikiHow to PDDL.ipynb (https://colab.research.google.com/github/interactive-fiction-class/interactive-fiction-class.github.io/blob/master/homeworks/planning/Example_of_How_To_Convert_WikiHow_to_PDDL.ipynb)
- Colab to Annotate Your PDDL with WikiHow Mentions.ipynb (https://colab.research.google.com/github/interactive-fiction-class/interactive-fiction-class.github.io/blob/master/homeworks/planning/Annotate_Your_PDDL_with_WikiHow_Mentions.ipynb)

Homework 4: Convert WikiHow to PDDL

Instructions

In this homework assignment, you will translate a wikiHow article into the Planning Domain Definition Language (PDDL) format that we explored in our in-class activity on planning and PDDL (https://interactive-fiction-class.org/in_class_activities/planning/planning-and-pddl.html). We'll also ask you to write your thoughts about how large LMs could be used to automatically generate plans from wikiHow-style instructions.

Here's an overview of what you'll do:

1. Pick a wikihow article about a task that might be interesting in a game, or just interesting to you.
2. Traslate some of the steps into PDDL.
    - A domain definition for the wikihow article that you picked
    - Schema for actions with pre-requisite and post conditions for each action
    - Several problem files that each define a goal corresponding to a step in the wikiHow article, and an initial state that is solvable with your action schemata in several steps.
3. Create annotations that link natural language descriptions in the wikiHow article text to the elements in your PDDL domain. You'll save this into a JSON file that you or your classmates might re-use for in your term projects.
4. Write a report that describes what you did. Your report should also describe what you think the limitations of the PDDL representation language are. You should also discuss how you might be able to use OpenAI to help convert a wikiHow article into PDDL. What would your inputs and outputs be? What data would you need in order to fine-tune a system?

Step 1: Pick an Interesting wikiHow Article

We'll use create our PDDL files from a wikiHow article. The goal for this is to start from something that describes proceedures and actions and is written in natural language, and then to manually translate it into the description language used for automated planning.

Here are a few wikiHow articles that I thought might be interesting since they had some elements that could make for interesting interactive fiction. It's fine to pick your own article. We won't translate the whole article, just a few steps, so you can pick out the parts that you think are most relevant / easiest to create action schema from.

Survival Stories

- How to Survive in the Woods (https://www.wikihow.com/Survive-in-the-Woods)
- How to Survive in the Jungle (https://www.wikihow.com/Survive-in-the-Jungle)
- How to Survive on a Desert Island (https://www.wikihow.com/Survive-on-a-Desert-Island)
- How to Survive on a Deserted Island With Nothing (https://www.wikihow.com/Survive-on-a-Deserted-Island-With-Nothing)
- How to Get Out of Quicksand (https://www.wikihow.com/Get-Out-of-Quicksand)
- How to Open a Coconut (https://www.wikihow.com/Open-a-Coconut)
- How to Test if a Plant Is Edible (https://www.wikihow.com/Test-if-a-Plant-Is-Edible)
- How to Find True North Without a Compass (https://www.wikihow.com/Find-True-North-Without-a-Compass)
- How to Survive a Wolf Attack (https://www.wikihow.com/Survive-a-Wolf-Attack)

Kid Detectives

- How to Make a Detective Kit (https://www.wikihow.com/Make-a-Detective-Kit)

- How to Disguise Yourself (https://www.wikihow.com/Disguise-Yourself)
- How to Make a Hidden Camera (https://www.wikihow.com/Make-a-Hidden-Camera)
- How to Hide Money (https://www.wikihow.com/Hide-Money)
- How to Spy on People (https://www.wikihow.com/Spy-on-People)
- How to Hack (https://www.wikihow.com/Hack)
- How to Make a Grappling Hook (https://www.wikihow.com/Make-a-Grappling-Hook)
- How to Open a Locked Door (https://www.wikihow.com/Open-a-Locked-Door)
- How to Create a Secret Society (https://www.wikihow.com/Create-a-Secret-Society)
- How to Win Fights at School (https://www.wikihow.com/Win-Fights-at-School)

Dystopian Futures

- How to Survive a Comet Hitting Earth (https://www.wikihow.com/Survive-a-Comet-Hitting-Earth)
- How to Survive an EMP (https://www.wikihow.com/Survive-an-EMP)
- How to Survive a Nuclear Attack (https://www.wikihow.com/Survive-a-Nuclear-Attack)
- How to Build a Fallout Shelter (https://www.wikihow.com/Build-a-Fallout-Shelter)
- How to Survive a Riot (https://www.wikihow.com/Survive-a-Riot)
- How to Survive Under Martial Law (https://www.wikihow.com/Survive-Under-Martial-Law)
- How to Avoid Danger During Civil Unrest (https://www.wikihow.com/Avoid-Danger-During-Civil-Unrest)
- How to Thwart an Abduction Attempt (https://www.wikihow.com/Thwart-an-Abduction-Attempt)
- How to Make Papyrus (https://www.wikihow.com/Make-Papyrus)

Step 2: Convert the Task to PDDL

As an example, I'll pick the How to Survive in the Woods (https://www.wikihow.com/Survive-in-the-Woods) article, and work on translating Part 1, Step 1 into PDDL. Here is step 1 from that article:

Finding Drinking Water

Search for a source of fresh water.[1] The first thing that you'll need in order to survive in the woods is water that you can drink. Look for signs of fresh water nearby like areas of green foliage that indicate water is nearby, low-lying areas where water could be collected, and signs of wildlife like animal tracks. It could mean that a creek, stream, or pond is nearby. While finding water is important for survival, be aware some water sources will not be safe - if possible treat all drinking water before using it. [2] If there are mountains nearby, look for water collected at the foot of the cliffs.

- The presence of insects like mosquitoes and flies means that water is nearby.
- Water from heavily oxygenated water (such as from a big waterfall or rapids) typically is safer than that from a slow or still water source.
- Freshwater springs are typically safer water sources, although these can be contaminated by mineral or bacteria as well.
- Remember that all untreated water must be considered risky unless treated. Even crystal clear water can harbor diseases and be dangerous if consumed.

Domain and Problems

You should create PDDL files for:

1. The domain. You should have a single `domain.pddl` file which defines the domain, including the types, predicates and action schemata that are relevant to your wikiHow article.
2. One or more problems. You should create one more problem file that defines a problem, an initial state, and a goal, that can be reached by applying the action schema that you defined. In some cases, it might make sense to have one problem file for each step in a wikiHow article.

The name of your domain should be the aricle title. I have started defining a domain for `survive_in_the_woods` .

```
(define (domain survive_in_the_woods)
   (:requirements :strips :typing)
   ...
```

I'll create a problem file for Step 1 in the article. I'll name the problem `collect_water` and I'll define the goal as `(:goal (and (inventory npc water))` where the player (or an AI-controled non-playable character, abbreviated as NPC) needs to add water to their inventory.

Here's what the start and end of the problem PDDL file would look like:

```
(define (problem collect_water)
   (:domain survive_in_the_woods)
...
   (:goal (and (inventory npc water)))
```

Types

You should specify what the `types` are in your domain. They should be the things that are relevant to solving the problem that you're tackling. It's sometimes also useful to create subtypes. Which you can do like this:

```
   (:types
       water - item
       player direction location
   )
```

By default all types are subtypes of `object` . The line

```
water - item
```

allows us to define `water` as a subtype of `item` (`item` is also introduced as a type in this same line). Having `water` as a subtype allows us to restrict some action schemas to only operate on that type.

Action Schema

Here's an example of an action schema for getting items:

```
(:action get
   :parameters (?item - item ?p - player ?l1 - location)
   :precondition (and (at ?p ?l1) (at ?item ?l1))
   :effect (and (inventory ?p ?item) (not (at ?item ?l1)))
)
```

It has the effect or removing the item from the current location and putting it into the player's inventory.

We might add a seperate action for getting water, since when our player gets some water from a lake, the water shouldn't be removed from that location. Here's one way to write it:

```
(:action get_water
   :parameters (?p - player ?loc - location ?water - water)
   :precondition (and (at ?p ?loc) (has_water_source ?loc))
   :effect (and (inventory ?p ?water) (not (treated ?water)))
)
```

(We might also consider adding some additional preconditions, like that the player has a container to store their water in).

Predicates

In your PDDL domain file, please define your predicates like this:

```
(:predicates
   (has_water_source ?loc - location) ; this location has a source of fresh water.
   (treated ?water - water) ; True if the water has been decontaimated by boiling it
   ...
)
```

You should give the type of each predicate's arguments, and a comment about what the predicate means (everything after the `;` is a comment).

Example Domain

Here's an example of how I started the `survive_in_the_woods` domain.

```
(define (domain survive_in_the_woods)
   (:requirements :strips :typing)
   (:types
       water - item
       player direction location
   )

   (:predicates
      (has_water_source ?loc - location) ; this location has a source of fresh water.
      (treated ?water - water) ; True if the water has been decontaimated by boiling it
      (at ?obj - object ?loc - location) ; an object is at a location
      (inventory ?player ?item) ; an item is in the player's inventory
      (connected ?loc1 - location ?dir - direction ?loc2 - location) ; location 1 is connected t
o location 2 in the direction
      (blocked ?loc1 - location ?dir - direction ?loc2 - location) ; the connection between loca
tion 1 and 2 in currently blocked
   )

   (:action go ; navigate to an adjacent location
      :parameters (?dir - direction ?p - player ?l1 - location ?l2 - location)
      :precondition (and (at ?p ?l1) (connected ?l1 ?dir ?l2) (not (blocked ?l1 ?dir ?l2)))
      :effect (and (at ?p ?l2) (not (at ?p ?l1)))
   )

   (:action get ; pick up an item and put it in the inventory
      :parameters (?item - item ?p - player ?l1 - location)
      :precondition (and (at ?p ?l1) (at ?item ?l1))
      :effect (and (inventory ?p ?item) (not (at ?item ?l1)))
   )

   (:action get_water ; get water from a location that has a water source like a lake.
      :parameters (?p - player ?loc - location ?water - water)
      :precondition (and (at ?p ?loc) (has_water_source ?loc))
      :effect (and (inventory ?p ?water) (not (treated ?water)))
   )
)
```

Example Domain

Here's how I specified the problem of collecting water from a source like a waterfall.

I instantiated several objects, and specified an initial state with a goal of of `(inventory npc water)`.

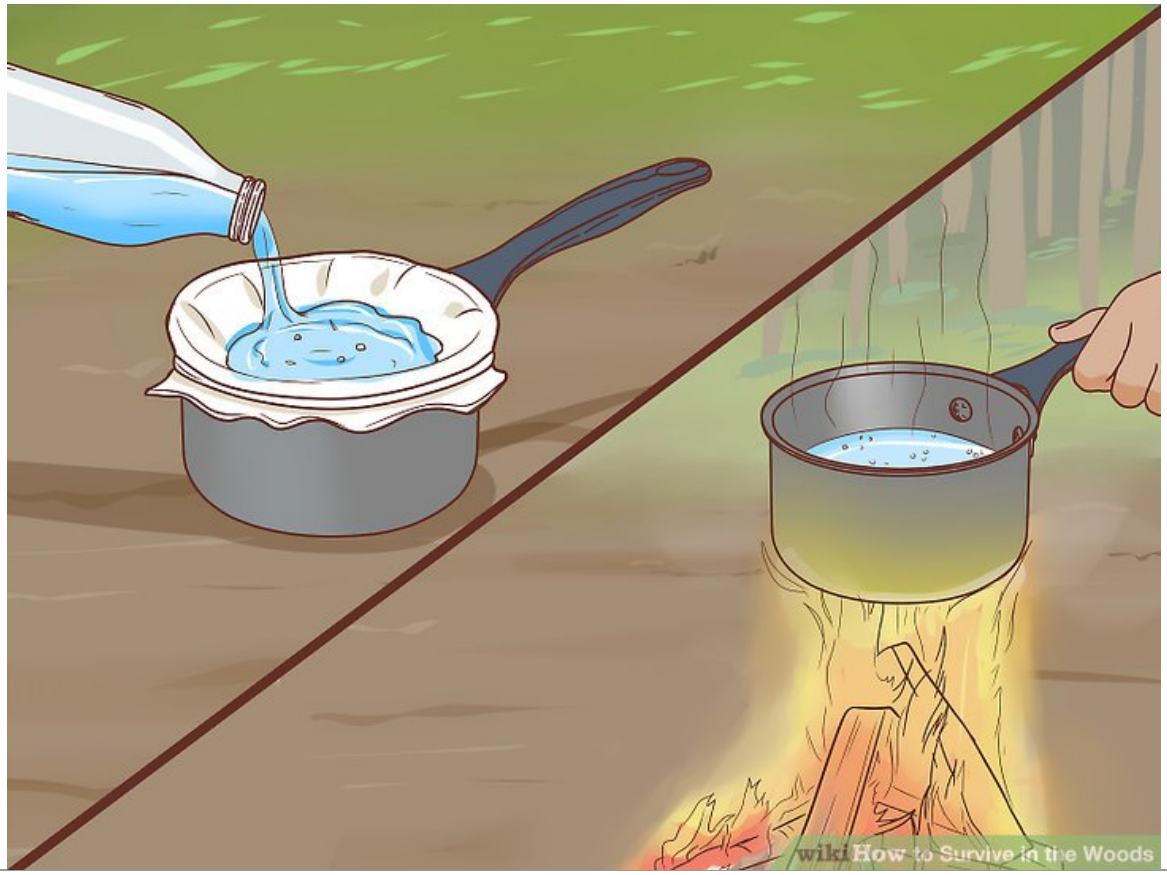
```
(define (problem collect_water)
   (:domain survive_in_the_woods)

   (:objects
      npc - player
      waterfall camp path cliff - location
      in out north south east west up down - direction
      water - water
   )

   (:init
      (connected camp west path)
      (connected path east camp)
      (connected path west cliff)
      (connected cliff east path)
      (connected cliff up waterfall)
      (connected waterfall down cliff)
      (at npc camp)
      (at canteen camp)
      (has_water_source waterfall)
   )

   (:goal (and (inventory npc water)))
)
```

Here's a sequence of actions that reaches the goal:

```
go west npc camp path
go west npc path cliff
go up npc cliff waterfall
get_water npc waterfall water
```

Other problems

If the player wants to survive in the woods, other problems remain. For starters, their water still isn't safe to drink! To fix that problem we could implement Step 6 of How to Survive in the Woods (https://www.wikihow.com/Survive-in-the-Woods):



Purify any water that you find.

It's extremely important that you purify any water that you collect, including rainwater, dew, and ice or snow, so you don't consume bacteria that could make you ill or even kill you. Use a piece of cloth or clothing to strain the water to remove large particles, then boil the water for 10 minutes to kill any contaminants.

- If you don't have a container to boil water in, you can fill a clear plastic bottle with water, seal the lid, and place the bottle on its side in direct sunlight for 6 hours to purify it.
- In the event that you have no containers and no way to purify water, you can dig a deep hole, let it fill with groundwater, and wait for the particles to settle at the bottom and the water is clear before you drink it. You should only do this if you have no other option.

I won't give the details of how to implement this, but you can imagine extending our `survive_in_the_woods` domain to add action schema for `strain`, `boil`, and `purify_in_sunlight`, and to create a new PDDL problem for `purify_water` with the goal `(:goal (and (inventory npc water) (treated water)))`.

That in turn might require us to solve a problem like `create_a_fire` in order to boil the water.

Step 3: Create Annotations for your Domain.

As a final step, you will annotate data and save a JSON file that links the phrases in the wikiHow article that you selected with the different elements of your PDDL elements.

If anyone is interested in doing a term project focused on automatically converting wikiHow to PDDL, then we'll share this JSON data with your classmates.

What to submit

You should submit the following:

- A set of PDDL files, one PDDL file for your domain, and one PDDL file for each of the problems. Your domain definition should have
    1. at least 10 action schemas beyond the ones that we defined for action castle,
    2. relevant types for your problem,
    3. predicates defined with their types and comments describing them.
- Your problem definitions should
    1. cover at least 3 problems,
    2. give initial states and goals for each, and
    3. ensure that the goal can be reached from the initial state using your action schema.
- A JSON file containing your annotations that map from the elements in your PDDL domain onto phrases in the wikiHow article that you selected. You can use this Colab to annotate your PDDL elements with mentions from your WikiHow article (https://colab.research.google.com/github/interactive-fiction-class/interactive-fiction-class.github.io/blob/master/homeworks/planning/Annotate_Your_PDDL_with_WikiHow_Mentions.ipynb).
- A PDF file containing your writeup. You should include at least 1 paragraphs for each of the following topics:
    1. What wikiHow article did you pick and why?
    2. What portions of the article did you select to translate to PDDL?
    3. Give some example of the actions, types, and predicates you used in your domain.
    4. Explain what goal you selected for your problem, and give the inital state and solution that you created.
    5. What limitations of PDDL did you encounter that makes it difficult to precisely convert a wikiHow description into PDDL?
    6. Could your PDDL be used as an interesting challenge for a text-adventure-style game? If so, how? If not, what would needed to create an interesting challenge?
    7. Discuss how you might use GPT-3 to automatically or semi-automatically convert a wikiHow article to PDDL?

Submissions should be done on Gradescope (https://www.gradescope.com/courses/354158/assignments/1882338).

Recommended readings

- Peter Norvig and Stuart J. Russell, AIMA Chapter 11 “Automated Planning” (https://www.amazon.com/Artificial-Intelligence-A-Modern-Approach/dp/0134610997).
- Planning.Wiki - The AI Planning & PDDL Wiki, PDDL Domain (https://planning.wiki/ref/pddl/domain).
- Planning.Wiki - The AI Planning & PDDL Wiki, PDDL Problem (https://planning.wiki/ref/pddl/problem).
