CIS700 HW4 Convert WikiHow to PDDL

March 1st, 2022

1 Convert wikiHow Article

1.1 Which article? Why?

We chose the wikiHow article "How to Survive in the Woods" to convert to PDDL because it contains tangible and realistic actions that can be taken, which is closely related to the action-castle style text-adventure game we have played. It is also useful to learn various survival tips something in case we need to survive in the woods some day.

1.2 Which portions were translated?

We translated **Part 4 - Starting a Fire**, which consists of six distinct steps that we will convert to PDDL-style problems, namely:

1. Find small, dry material to use as tinder
2. Gather small twigs and branches to use for kindling
3. Collect larger logs for fuel
4. Build a teepee structure with the tinder and kindling
5. Create a fire plough to ignite the kindling and start the fire
6. Use a fire to keep warm, cook your food, and purify your water

1.3 Examples in our domain

1.3.1 Types

We used the following types in our domain definition. The majority of them, such as grass, branch, and log, are subcategories of "item" and they represent objects that we need to gather in order to accomplish certain actions specified by the wikiHow article. We also have types "player", "direction", and "location". They are ported from the action-castle PDDL we did for the in-class exercise. Since one of our motivations is to examine if PDDL can be applied to text adventure games, we framed our domain to include a player (which enables us to check if a player has an item in their inventory or if the player is in the right location), location (which sets up a world with different attributes and available actions), and direction (which allows the player to navigate the world).

```
(:types
   grass branch log tinder kindling fuel teepee wood water pot - item
   player direction location
)
```

1.3.2 Predicates

Here are some sample predicates we used in our domain. They allow us to keep track of states of the problem as well as determine pre-conditions of tasks we need to perform. For example, to gather logs for fuel the player needs to be at a location where there are dry oak trees. Hence the "has_dry_oak" predicate.

```
(:predicates
  (has_dry_oak ?loc - location)          ; this location has dry oak trees
  (has_pond ?loc - location)             ; this location has a pond with water
  (dry ?item)                            ; True if the item is dry
  (ignited ?item)                        ; True if the item is ignited
  (inventory ?player ?item)              ; an item is in the player's inventory
)
```

1.3.3 Actions

Here are two sample actions we used in our domain - "get_log" and "stack_log". They make up the third step "collect larger logs for fuel" in the wikiHow article we chose. For the "get_log" action, the precondition requires that the player is present at a location where there are dry oak logs and the effect of this action is that the player would have oak logs in their possession or inventory. Then the action "stack_log" takes the oak logs in the player's inventory to make a log stack.

```
(:action get_log ; log oak trees
  :parameters (?p - player ?loc - location ?oak_log - log)
  :precondition (and (at ?p ?loc) (at ?oak_log ?loc) (dry ?oak_log))
  :effect (and (inventory ?p ?oak_log))
)

(:action stack_log ; stack dry logs to make enough fuel
  :parameters (?p - player ?oak_log - log ?log_stack - fuel)
  :precondition (and (inventory ?p ?oak_log))
  :effect (and (inventory ?p ?log_stack) (not (inventory ?p ?oak_log)))
)
```

1.3.4 Problem Example

Here we list the types, predicates, and actions that are directly relevant to solving the problem of "collect kindling" in our domain. The reasons for such design are explained in item 2 in subsection 1.4.

```
(:types
   ... branch kindling ... - item
   player direction location
)

(:predicates
  (has_large_dry_branch ?loc - location) ; this location has large and dry
    branches that can be broken off as kindling material.
)
```

```
(:action break_branch ; break larger branches into smaller pieces
  :parameters (?p - player ?loc - location ?branch_piece - kindling)
  :precondition (and (at ?p ?loc) (has_large_dry_branch ?loc))
  :effect (and (at ?branch_piece ?loc))
)
```

1.4 Goal, Initial States, and Solution

The following initial states are attributes of the world that are consistent for all problem in our domain. We took inspiration from action-castle and created a "map" for our wikiHow article. Each location has different attributes that makes it unique. For example, we can only obtain the item "grass" from location "grassland". The player would also always start at the camp.

```
(:objects
    npc - player
    camp path grassland bush oaks wilderness - location
    in out north south east west up down - direction
)

(:init
    (connected camp west path)
    (connected path east camp)
    (connected path north grassland)
    (connected grassland south path)
    (connected grassland north bush)
    (connected bush south grassland)
    (connected path west wilderness)
    (connected wilderness east path)
    (connected path south oaks)
    (connected oaks north path)
    (at npc camp)
)
```

1. Find small, dry material to use as tinder

```
(:objects
    ...
    grass dry_grass - grass
    grass_tinder - tinder
)

(:init
    ...
    (at grass grassland)
)

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

```
PDDL Solution Plan:
    go west npc camp path
    go north npc path grassland
    get grass npc grassland
    cluster_grass npc grass grass_tinder
```

2. Gather small twigs and branches to use for kindling (used in 1.3.4).

```
(:objects
    ...
    branch_piece - kindling
)

(:init
    ...
    (has_large_dry_branch bush)
)

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

The goal of this problem is to get some kindling. However, the original branch was too large to use, so we needed to break it down into pieces.

```
PDDL Solution Plan:
    go west npc camp path
    go north npc path grassland
    go north npc grassland bush
    break_branch npc bush branch_piece
    get branch_piece npc bush
```

3. Collect larger logs for fuel.

```
(:objects
    ...
    axe - item
    oak_log - log
    log_stack - fuel
)

(:init
    ...
    (at axe camp)
    (has_dry_oak oaks)
)

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

```
PDDL Solution Plan:
    get axe npc camp
    go west npc camp path
    go south npc path oaks
    log_oak npc oaks axe oak_log
    get oak_log npc oaks
    stack_log npc oak_log log_stack
```

4. Build a teepee structure with the tinder and kindling.

```
(:objects
    ...
    teepee_structure - teepee
)

(:init
    ...
    (has_dry_even_area camp)
)

(:goal (and (at teepee_structure camp)))
```

```
PDDL Solution Plan:
    go west npc camp path
    go north npc path grassland
    go north npc grassland bush
    break_branch npc bush branch_piece
    get branch_piece npc bush
    go south npc bush grassland
    get grass npc grassland
    go south npc grassland path
    go south npc path oaks
    cluster_grass npc grass grass_tinder
    log_oak npc oaks branch_piece oak_log
    get oak_log npc oaks
    go north npc oaks path
    go east npc path camp
    stack_log npc oak_log log_stack
    build_teepee npc camp grass_tinder branch_piece log_stack teepee_structure
```

5. Create a fire plough to ignite the kindling and start the fire.

```
(:objects
    ...
    flat_wood - wood
)

(:init
    ...
    (at flat_wood oaks)
)

(:goal (and (ignited teepee_structure)))
```

```
PDDL Solution Plan:
    go west npc camp path
    go north npc path grassland
    go north npc grassland bush
    break_branch npc bush branch_piece
    get branch_piece npc bush
    get_branch npc bush branch
    go south npc bush grassland
    get grass npc grassland
    go south npc grassland path
    go south npc path oaks
    get flat_wood npc oaks
    cluster_grass npc grass grass_tinder
    log_oak npc oaks flat_wood oak_log
    get oak_log npc oaks
    go north npc oaks path
    go east npc path camp
    stack_log npc oak_log log_stack
    build_teepee npc camp grass_tinder branch_piece log_stack teepee_structure
    ignite_wood npc camp flat_wood branch teepee_structure
    ignite_teepee npc camp flat_wood teepee_structure
```

6. Use a fire to purify your water.

```
(:objects
    ...
    pot - pot
```

```
    water - water
)

(:init
    ...
    (at pot camp)
    (has_pond grassland)
)

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

```
PDDL Solution Plan:
    get pot npc camp
    go west npc camp path
    go north npc path grassland
    go north npc grassland bush
    break_branch npc bush branch_piece
    get branch_piece npc bush
    get_branch npc bush branch
    go south npc bush grassland
    get grass npc grassland
    cluster_grass npc grass grass_tinder
    get_water npc grassland water pot
    go south npc grassland path
    go south npc path oaks
    get flat_wood npc oaks
    log_oak npc oaks pot oak_log
    get oak_log npc oaks
    go north npc oaks path
    go east npc path camp
    stack_log npc oak_log log_stack
    build_teepee npc camp grass_tinder branch_piece log_stack teepee_structure
    ignite_wood npc camp flat_wood branch teepee_structure
    ignite_teepee npc camp flat_wood teepee_structure
    boil_water npc camp teepee_structure water
```

.

2 Limitations of PDDL in Converting wikiHow

- Cost: It is too exhausting for humans to choose the necessary actions in logical order and translate their preconditions and effects in detail. The more complicated a problem or domain is, the worst it becomes.
- Diversity: It is hard to design multiple ways to achieve the same goal.
- Intuitiveness: PDDL is not very intuitive to follow or interpret logically. Simple tasks could be very convoluted when they are translated into the required schema.
- Debugging: PDDL almost has the complexity of a programming language but none of the essential debugging tools or methods. When translating large problems or domains by hand, mistakes are bound to happen. Small typos and spacing issue are extremely difficult to spot and the error messages might not be helpful at all.

3 Discussion

3.1 PDDL and Text-Adventure Game

Yes, we can design a plot where if the player accidentally enters the woods and gets lost, he needs to start a fire to survive the freezing night. That is when he or she would need our solution to achieve the goal. Since we incorporated different locations and directions in the world, this will allow the player to explore around the wilderness, thus creating interesting challenges in a text-adventure game.

3.2 PDDL using GPT-3

We have annotated our PDDL with wikiHow mentions in this homework. Along with the ones that our classmates have created, we can easily curate a decent training dataset with wikiHow mentions as prompts and PDDL as completion. Then use them to fine-tune a GPT-3 model. After that, we could either work with GPT-3 to design types, predicates, actions, or just let GPT-3 take over and do all the work.