9. Appendices

9.1. Prompts

Prompt without CoT:
task requirement:
could you fill out the below pddl actions with the predicates based on the text?
All fields: parameters, precondition and effect, should have predicates.
For each action, do NOT change the name and do NOT drop the action and do NOT add more actions.
The output should be in correct pddl format.

wikiHow text and domain header:
here are the actions I want:
<insert_action_names>
here are the requirements I have:
<insert_requirements> here are the types I have:
<insert_types>
here are the predicates I have:
<insert_predicates>
here are the texts containing steps to <insert_goal>:
<insert_text>

Example Completion:
(:action clean_water
:parameters (?player - human ?water - water)
:precondition (inventory ?player ?water)
:effect (treated ?water)
)

Prompt with CoT:
task requirement:
could you fill out the below pddl actions with the predicates based on the text? All fields: parameters, precondition and effect, should have predicates.
For each action, do NOT change the name and do NOT drop the action and do NOT add more actions and:
First, summarize the action in a few sentences based on the text and provide its requirements and its aims if it has.
Next, identify ALL the entities involved in the action and describe whether it changed, unchanged, added, removed in the action in natural language.
Last, translate it into pddl format. Check all the related entities are in the 'parameters'.

output format:
Please use this output format:
- action name: ...
- summarize action: ...
- what is need to do the action: ...
- what is the result: ...
- entities:
- entity name: ...
- before: ...
- after: ...
...

- describe how to match it to pddl relevant predicates step by step:
1. ...
2. ...

pddl:

wikiHow text and domain header:
here are the actions I want:
<insert_action_names>

here are the requirements I have:
<insert_requirements>

here are the types I have:
<insert_types>

here are the predicates I have:
<insert_predicates>

here are the texts containing steps to <insert_goal>:
<insert_text>

Example Completion:
- action name: clean_water
- summarize action: The player cleans water in their inventory using heat from a fire.
- what is needed to do the action: The player must have untreated water in their inventory and be at a location with fire.
- what is the result: The player has treated water in their inventory.

- entities:
- entity name: player
- before: Having untreated water in inventory.
- after: Having treated water in inventory.
- entity name: water
- before: Untreated.
- after: Treated.

- describe how to match it to pddl relevant predicates step by step:
1. Check if the player has untreated water in their inventory.
2. Check if the player is at a location with a fire.
3. Replace untreated water with treated water in the player's inventory in the effect.

pddl:

(:action clean_water
:parameters (?player - human ?loc - location ?water - water)
:precondition (and (at ?player ?loc) (inventory ?player ?water) (not (treated ?water)) (has_fire ?loc))
:effect (treated ?water)
)

9.2. WikiHow Topics

Below are a list of the wikiHow articles that were converted to PDDL by students in a graduate seminar course that focused on interactive fiction games.
create secret society
throw an anime party
open a coconut
calculate pi by throwing frozen hot dogs
hack
get out of quicksand
make a detective kit
lock picking
make papyrus
survive on a desert island
survive in the jungle
survive a war
survive a comet hitting earth
survive a nuclear attack
survive in the woods
survive deserted island
survive shark attack
survive emp attack

9.3. Ablation of Input Text

Input text description
We consider the following choice of wikiHow texts as $\mathbb{T}$.
Prompt without text (w/o $\mathbb{T}$) is an ablation baseline where the model predicts A solely based on H. Naturally, none of the three aforementioned stages are involved in this prompt condition.
Prompt with text (w/ $\mathbb{T}$) additionally provides the model with four different portions of $\mathbb{T}$, involving the three aforementioned stages, as follows:
($\mathbb{T}$ = all): All steps in a wikiHow article.
($\mathbb{T}$ = rel): In PROC2PDDL, each wikiHow article consists of step paragraphs that may or may not be used in defining the actions in the $\mathbb{DF}$. Hence, a mapping between actions and steps is also annotated. This context includes elevant steps to all actions in a $\mathbb{DF}$.
(e.g., Step 1. Find fresh water... Step 2. Collect food... Step 7. Set up camp...)
($\mathbb{T}$ = map): Each action mapped with steps based on the annotated mapping in PROC2PDDL.
(e.g., clean_water: Step 1. Find fresh water...)

Model %	Intrinsic action acc.	Extrinsic $\mathbb{PF}$ solve	Extrinsic exact plan
w/o $\mathbb{T}$ (baseline)	13.7	26.3	3.2
$\mathbb{T}$ =sum	15.9	33.7	4.2
$\mathbb{T}$ =sum, CoT	**18.1**	**35.8**	**6.3**
$\mathbb{T}$ =map	11.8	13.7	2.1
$\mathbb{T}$ =map, CoT	8.9	26.3	1.1
$\mathbb{T}$ =rel	11.6	27.4	0.0
$\mathbb{T}$ =rel, CoT	12.2	21.1	4.2
$\mathbb{T}$ =all	12.1	28.4	0.0
$\mathbb{T}$ =all, CoT	12.1	31.6	0.0

Table 6: Performance using different portions of text $\mathbb{T}$. Metrics include action-wide accuracy, average edit distance of action definitions, the proportion of $\mathbb{PF}$s that can be solved, and the proportion of generated plans that exactly match the gold plans.

($\mathbb{T}$ = sum): One-line summaries of actions annotated in PROC2PDDL.
(e.g., clean_water; boil water to clean it)
The four prompts are increasingly general. Distinguishing from the requirements, the first text conditions demand accurate information identification and extraction, while the last one clearly defines the action but still requires the model to infer implicit entity states. All prompts request an exact translation.

Result analysis
In w/o text setting fully relying on its implicit knowledge, the model is already capable of inferring PDDL syntactically and semantically. In w/ text settings, our model shows an 'U' performance in terms of the text length. Using a sentence-long description for each action provided by PROC2PDDL, the model achieves the best performance among all, showing a strong deduction ability with the limited but precise NL input. The **`all`** setting ensues, which requires the most extraction rather than inference. In contrast, the middle ones (**`rel`**/**`map`**) with decreasing signal-to-noise ratio lead to worse results, indicating its shortage of extraction-inference trade-off. The signals contain both the described entity states and step relations, explicitly and implicitly. This shortage may come less from the entity states (e.g., fish, spear in hunt_fish), but more from the relation between actions (e.g., make_spear to hunt_fish) which may be expressed in the **`sum`** and **`all`** settings.

9.4. Calculating Actions Equivalence

The distance between two actions can be divided to two parts:

1. The distance between parameters:

   We don't need to care about the specific parameter names; we only need to consider the

parameter types. For each parameter in Action1, we iterate over the parameter list of Action2 to find the first parameter in Action2 with the same type. We use two hash maps, p1 and p2, to record these two parameters and their corresponding types. We increment the counter by 1, remove that parameter from the parameter list of Action2, and break from the current loop. After the iteration, we obtain the number of matching parameters, n. The distance between parameters can be calculated as |number of parameters in Action1 $-n|+$ |number of parameters in Action2 $-\ n|$.

2. The distance between preconditions/effects:

   For each condition in Action1, we iterate over the condition list of Action2. The conditions can only match if they have the same predicate and the same number of parameters. We iterate over the parameters in these conditions and make the following judgments:

   - If neither of the two current parameters has appeared before (in p1 and p2) and these parameters are not identical, they don't match.
   - If the two parameters have different categories, they don't match.
   - If the two parameters have the same categories and don't have an index, we consider them as the same parameter entity and give them the same index. We continue the iteration.
   - If the two parameters already have indexes, we check if the indexes are equal. If they are not equal, they don't match. Otherwise, we continue the iteration.
   - In any other case, they don't match.

   If all parameters of the two conditions match, we increment n by 1. The distance between preconditions/effects can be calculated as |number of preconditions/effects in Action1 $-\ n|+$|number of preconditions/effects in Action2$-\ n|$.