Interactive Fiction and Text Generation, CIS 700
Homework # 4 Write-up

March 1, 2022

1 What WikiHow Article

We choose the article **How to Hack.**[1] In short, we pick this one because it is cool. Besides, we think the process of hacking is easier to be converted into a routine because the objects and tools are more or less well-defined. For instance, *Python* is not ambiguous in the context of hacking: we all know it refers to the programming language, not the wild animal. Also, there are automated hacking (or testing) softwares out there which proves that the hacking procedures can be formalized.

2 What Portion

The article is composed of two parts: *Learning the Skills Needed to Hack*, and *Hacking*, where the second part is the actual "implementation" of hacking while the first part is foundation-building. So, we implemented both of them. We formalized the first part into a single problem and divided the second part into two problems.

3 Example Domain

Intuitively, we can decompose a hacking process into several "landmarks" or "stages". For example, getting root password can be one stage (which is pretty close to the end). So, we formalize these stages as `location`. In the problem code we have,

```

        ; "Problem 1 (Part I), learn hacking basics"
        (:types
            rookie   - location  ; "Init stage: knows nothing about hacking"
            skilled  - location  ; "Goal stage: has built a solid foundation"
        )

```

Again, take the first problem, learning hacking basics as an example. The player or NPC should focus on learning all kinds of skills and concepts during this stage. In this case, we also defined several types for corresponding skills and predicates to denote whether a player has mastered the required skills.

[1] https://www.wikihow.com/Hack

```
1
2       ; "Problem 1 (Part I), learn hacking basics"
3
4       ; "Some skill types"
5       (:types
6           cpp      - skill
7           py       - skill
8           php      - skill
9           ; ...
10          hacking  - concept
11          ethics   - concept
12      )
13
14      ; "Corresponding predicates"
15      (:predicates
16          (learned          ?p - player ?s - skill)
17          ; ...
18          (know_programming ?p - player)
19          (know_hacking     ?p - player)
20          (know_ethics      ?p - player)
21      )
22
23      ; "Actions denoting learning"
24      (:action learn
25          :parameters     (?p - player ?s - skill)
26          :precondition   (not (learned ?p ?s))
27          :effect         (learned ?p ?s)
28      )
29
```

Before a player can proceed from one hacking stage to another, a "block" needs to be removed. We initialize the state such that stages are blocked, e.g., `(blocked rookie skilled)`. Once certain precondition have been met, an action can be invoked to remove the blocks. Then, the player can `go` from one stage to another.

```
1
2       ; "Problem 1 (Part I), learn hacking basics"
3       (:action become_skilled
4           :parameters (
5               ?p      - player
6               ?l1     - location
7               ?dir    - direction
8               ?l2     - location)
9           :precondition (and
10              (know_programming ?p)
11              (know_hacking     ?p)
12              (know_ethics      ?p)
13              (know_internet    ?p)
14              (know_unix        ?p))
15          :effect (not (blocked ?l1 ?dir ?l2))
16      )
17
```

While hacking, we formalize the tools for hacking and various credentials as `item` gettable at different hacking stages (`location`). The rest of the intuitions are pretty much the same.

4 Goal, Initial State, and Solution Found

As mentioned above, we formalize the entire hacking process as a series transitions between hacking stages. Therefore, the goals of the three sub-problems are `(at npc skilled)`, `(at npc`

`knowing_target)`, and `(at npc end_hacking)`, respectively. The initial states and solutions found are listed as follows.

4.1 Hacking Basics

We expect NPC to learn various skills and concepts from the "environment", a set of skills initialized, and become skilled.

```

    (define (problem build_foundation)
        (:domain how_to_hack)
        (:objects
            ; ...
            ; "hacking states"
            rookie    - location
            skilled   - location

            ; "skills and concepts"
            c_hacking - hacking
            s_cpp     - cpp
            ; ...
        )

        (:init
            (at npc rookie)
            (connected rookie next skilled)
            (blocked   rookie next skilled)
        )

        (:goal (at npc skilled))
    )

```

As expected, solutions are a series of learning processes.

```

    understand npc c_ethics
    understand npc c_hacking
    learn npc s_unix
    learn npc s_asb
    learn npc s_php
    learn npc s_py
    learn npc s_cpp
    learn npc s_html
    learn npc s_search
    learn_concepts npc c_hacking c_ethics
    learn_internet npc s_html s_search
    learn_programming npc s_cpp s_php s_py s_bash s_asb
    learn_unix npc s_unix
    become_skilled npc rookie next skilled
    go next npc rookie skilled

```

4.2 Knowing Target

This is a very simple step: the player just need to get a permission at the initial state `securing_machine`. And we initialize the permission to that state instead of describing hand-by-hand steps of how to acquire such a permission, which is not included in the WikiHow article either.

```
(define (problem hacking1)
    (:domain how_to_hack)
    (:objects
        ; ...
        securing_machine - location
        knowing_target   - location
        white_permission - permission
    )

    (:init
        (connected securing_machine next knowing_target)
        (blocked securing_machine next knowing_target)
        (at npc securing_machine)
        (at white_permission securing_machine)
    )

    (:goal (at npc knowing_target))
)
```

The solution is very straightforward because the `machine_secured` action is very comprehensive.

```
get npc white_permission securing_machine
machine_secured npc white_permission securing_machine next knowing_target
go next npc securing_machine knowing_target
```

4.3 Hack'em

The code describing this step is quite trivial. The player use a certain tool at hand to get access to some credentials; use that credential to proceed to next stage; and repeat the process. We initialized various credentials at different stages for the player to find. Therefore, the code or solution is not listed here.

5 PDDL Limitations

We primarily find **Controlling Granularity** very challenging. In other words, how detailed an action should be? For instance, one step in the article is to *open a port*. But, open a port itself is a comprehensive action that can be decomposed into many sub-actions. One heuristic solution to this is to do-as-the-article-says. That is, if the original text is *open a port*, then we define an action `open_a_port`. But this raise a serious problem of **ambiguity.** Even if we try to alleviate this problem by passing a lot parameters to actions or predicates, the functional programming style of PDDL makes it very hard for us to track the states and objects floating around.

6 PDDL for Text-adventure-style Game

We think one potential use case is for a built-in helper, or hints. When a player struggles completing certain goal, such as finding certain item, we may allow the user to specify a goal, and let the helper dynamically generate a sequence of high-level procedures given the current state of player and the state of environment. Since a human user does not have difficulty handling ambiguity, the problem mentioned above can be mitigated.

7 WikiHow to PDDL by GPT-3

We believe that fully automated, end-to-end procedure "translation" is not feasible. Instead, we do think we can predefine a set of **atomic actions**, just like a vocabulary. Then, we may be able to formalize PDDL generation as a translation problem. So, we can let GPT-3 to conditionally select one atomic action at each time step, conditioning on the WikiHow article and all previously generated actions, predicates, and items.