Declan Kelly - G00378925

I have chosen to create an implementation of Tower of Hanoi. The objective is to transfer all the disks one by one from the starting rod to the right side while keeping the original order of the disks. But, there is a catch, you can only move a disk to a rod if there is no other disk on that rod or if the disk on the rod you are moving to is larger than the disk being moved. You must move disks between the other rods to solve the problem.

Upon the game starting, you will be presented with 3 rods, with the first one having all the eight disks. Once you have completed the goal of moving all the disks to the right-side, you will be presented with a "You won!" in yellow. As seen in the `Demonstration.mp4`.

Grammar

The **MOVE_DISK_BY_COLOUR** and **MOVE_FROM_ROD_TO_ROD** rules, use the **COLOURS** sub-rule.

```
<rule id="COLOURS">
  <one-of>
    <item><item>RED</item><tag> out = RED;</tag></item>
    <item><item>GREY</item><tag> out = GREY;</tag></item>
    <item><item>GREEN</item><tag> out = GREEN;</tag></item>
    <item><item>YELLOW</item><tag> out = YELLOW;</tag></item>
    <item><item>CYAN</item><tag> out = CYAN;</tag></item>
    <item><item>BLACK</item><tag> out = BLACK;</tag></item>
    <item><item>MAGENTA</item><tag> out = MAGENTA;</tag></item>
    <item><item>WHITE</item><tag> out = WHITE;</tag></item>
  </one-of>
</rule>
```

As seen in the **MOVE_DISK_BY_COLOUR** and **MOVE_FROM_ROD_TO_ROD** rules, they call the **COLOURS** rule, and extract the `out`, with `rules.latest()`.
The source colour is put in `fromColour` and the destination is put in `toColour`. The C# code will extract these from the semantic meanings.

```
<!-- RULE THREE -->
<rule id="MOVE_DISK_BY_COLOUR">
<tag>out.action = "MOVE_DISK_BY_COLOUR";</tag>
<item>MOVE</item>

<item><ruleref uri="#COLOURS"/></item>
<tag>out.fromColour = rules.latest();</tag>
<item>DISK</item>
<item>TO</item>

<item><ruleref uri="#COLOURS"/></item>
<tag>out.toColour = rules.latest();</tag>
<item>ROD</item>
</rule>
```

The filler/function words are also implemented in their own rule called **FUNCTION_WORDS**, the

rules below references this, for **THE** . . .
These are optional (using repeat 0-1) as seen in rule 1, `<item repeat="0-1"><ruleref uri="#FUNCTION_WORDS"/></item>`

1. ADD ROD

This will add a new rod to the right side of the game; a maximum of 8 rods are allowed. Alias: **APPEND ROD**, Functional filler words are included. **ADD A ROD**, **ADD THE ROD**, **APPEND A ROD** and **APPEND THE ROD**.

The aliases are implemented using `<one-of>`.

2. REMOVE ROD

This will remove a rod from the right side of the game, you can't remove a rod with disks on it, these will need to be removed first. Aliases: **DELETE ROD** and **DESTROY ROD**, Functional filler words are included like above.

3. MOVE X DISK TO Y ROD

X: {"RED", GREY, GREEN, YELLOW, CYAN, BLACK, MAGENTA, WHITE}
Y: {"RED", GREY, GREEN, YELLOW, CYAN, BLACK, MAGENTA, WHITE}

Same goes for X and Y in rule 4. If you are having problems moving a disk, make sure your adhering to the rules of the game.

Allows you to move a disk of colour X to a rod of colour Y.

4. MOVE FROM X ROD TO Y ROD

Allows you to move a disk from a rod of colour X to a rod of colour Y.

5. RESET THE GAME

This will reset the game back to its original state, resetting the move counter aswell.
Aliases for **RESET**, include: **ORDER** and **SORT**.

6. SHOW MOVE COUNT

This will display the amount of moves in the top left hand corner.
Aliases: **SHOW MOVE COUNT**, **DISPLAY MOVE COUNT**, **SHOW CHANGE COUNT** and **DISPLAY CHANGE COUNT**.

7. SHUFFLE THE GAME

This will mix up the state of the game, distributing the disks across the rods. Aliases for **SHUFFLE**, include: **MIX**, **RANDOM**, **RANDOMIZE** and **RANDOMISE**. You can use **STATE** instead of **GAME**.