FAQ:

Q: For the AI to get strength, it must **exchange 3 diamonds with a wizard**. What if the wizard already has the three diamonds? Is it necessary for the AI to trade for the three and then give them back or would the wizard just be able to give the AI the strength?

A: Think of the diamonds as payment. Thus the hero needs to give the diamonds as payment for the spell.

Q: I figured that I would make the hero strong through something like **(strong hero)**, but then I started wondering what specifically it should be. If the goal state required something like (isStrong hero), it would fail to find a plan even though clearly strong and isStrong are talking about the same thing. I checked in the test files, and this doesn't appear in any goal states.

In case it does in future test cases, what name should we give this condition?

A: Excellent question. Since I have not given any guidance, we will make sure that strong is not a goal predicate.

Q: Can **nonheroes trade** with each other? From the PDF, it sounds like they should be able to. On the other hand, I can't imagine a scenario where that would make the plan shorter, especially since they can't travel, so maybe it doesn't matter.

A: Non-heroes can trade.

Q: Does **killing the dragon remove its sleep** status? And does using the song require the dragon to be alive?

A dead dragon is not asleep. Killing it then means it is not asleep. Similarly, if the dragon is dead is can't go to sleep.

I note that I recently discovered that adding some of these specifications will make some plans too long. That is ok, the upper bounds on the plan length are a little arbitrary.

Q: Could **dragon** possess something and hero **trade** with him? I think dragon could possess sth. but hero is not allowed to trade with it.

A: Yes, if the dragon has something to trade, it can trade.

Q: Can we slay the dragon when he is asleep?

A: I have noticed that in some cases, the solver/planner returns a plan where the dragon is put to sleep and then it is slain. This is not an ideal or optimal plan, but it is correct since there is no requirement for the dragon to be awake when it is killed. Since this is not an optimal plan, it may go over the upper limit for length of the plan. Do not worry about this. We will be reasonable about the length of the plan since the planner is not optimal.

Additional note:

Be aware when defining predicates in your code that order of arguments matters. This is because the solver is very precise and does not understand semantics. To check if a predicate is valid at any point it applies the predicate to each state clause, such as `(possesses ai pointy)`, and if the clause matches exactly then the predicate succeeds. In this example if the `possesses` predicate says it takes first an `item` and then a `hero` it will fail to match the state clause with a type error.

A far more subtle error can arise with `different`. In `:init` the diamonds are specified different in specific pairings where the order matters.

```
(different d1 d2)
(different d2 d3)
(different d1 d3)
```

The solver recognizes that `(different d1 d2)` is not the same as `(different d2 d1)`. If you define a predicate `different` to have two objects as arguments, it will only succeed if there is a `different` clause in a state (in our tests the only way would be from `:init` but you could define an action that would generate a `different` clause in state) that exactly matches, including the order of the elements.