Reading 20: A Second Look at Prolog

# Exercise 1: Summarize

Prolog's unification uses substitutions to make terms identical through the Most General Unifier (MGU), explores procedural, implementational, and abstract execution models, discusses the occurs check, resolution, interpreter, proof trees, variable renaming, quoted atoms, I/O operations, assert and retract predicates, the cut, and illustrates these concepts with an adventure game example.

# Exercise 2: Demonstrate & Explain

Review the "adventure game" example at the end of the chapter. Identify code from the adventure game that illustrates occurrences of unification, substitution, backtracking, resolution, or proof trees. Create excerpts of the adventure game Prolog code, and write a sentence or two that justifies why the code relies on unification, substitution, backtracking, resolution, or proof trees. Alternatively, you may prepare and narrate a mini-presentation and/or video screen recording to communicate your demonstration.

### Unification:

* Make two terms identical with substitution
* Report predicate substitutes the X in at() with the Y in description.

report :-

at(you, X),

description(X, Y),

write(Y), nl.

### Substitution:

* Replace a variable with a term.
* Unification uses substitution, Y is replaced with “valley” and X is replaced by Y.

report :-

at(you, X),

description(X, Y),

write(Y), nl.

### Backtracking:

* Undo variable binding until desired result is reached.
* If the Move input fails then the program needs to backtrack the response from a move to writing this error message.

move(\_) :-

write('That is not a legal move.\n'),

report.

### Resolution:

* Derive a new clause by combining existing clauses.
* The main predicate is a collection of predicates that are called to run and check the status of the game.

main :-

write('\nNext move -- '),

read(Move),

call(Move),

ogre,

treasure,

cliff,

main.

### Proof trees:

* Steps necessary to prove a goal.
* Most predicates can use proof trees, the report predicate has a set goal (return report) and it goes through a few paths:

report

|----at(you, X)

| |----X = valley

|----description(X, Y)

| |----Y = '~some string~'

|----write(Y), nl

report :-

at(you, X),

description(X, Y),

write(Y), nl.

# Exercise 3: Inquire

Unification and Substitution are very similar, what is the main difference between them?

What kinds of predicates do not fall under Proof Trees? Show an example.