Backwards Chaining with First-Order Logic

What is backwards chaining?

- It is the idea of starting with the goal and working backward to validate it based on the knowledge base of facts and rule, which can be made of facts that have relations with each other
- For example, let's imagine you have a knowledge base with these facts and rules:
    - Facts: Alice is a parent of Tom, Tom is a parent of Charlie
    - Rules: if x is a parent of y AND y is a parent of z, then x is an ancestor of z
    - You want to prove that Alice is the ancestor of Tom.
    - The steps:
        - We have a matching rule that concludes that x is an ancestor of z.
        - We will check in the first condition of the rule if x is a parent of y (meaning that it does exist in the knowledge base).
            - Based on the rule concluding that x is an ancestor of z, we will know that x = Alice. Then once we find a fact, then we know that z = Tom.
        - Now that we know that z = Tom, and that Y = Charlie (we know this information based on the conclusion drawn from our AND rule created), and that the knowledge that Tom is a parent of Charlie exists in the knowledge base, proof that Alice is an ancestor of Tom is valid.
    - Backwards chaining will require substutuions since rules don't have to be used for one fact, but many.

Steps in doing a backwards chaining:

```
Validate_goal(goal, facts, rules):

//g will contain the main fact we want to prove.

For each fact, do a direct match against g. If a match was found
return True, otherwise keep going down in the validate function

For each fact, if the relationship type between the variables matches
with that of goal, then perform some unifications where we want to
substitute variables with their values. Example in fact f has
Parent(Alice, Tom) and goal g has Parent(x, y), so we would unify x
with Alice and y with Tom.

For each rule r, go through each of the sub-facts sb in r, and
recursively validate sb.
```