Functional Programming Project

Level 1

Level 1 is implemented in `LogicEvaluator.hs`. The program `evalProp` will check for each predicate if it is true or when it does not have a truth value and has a deriving right hand side rule, it will check for each predicate in that rule for a truth value or for a deriving right hand side.

Usage :

1. Run `ghci LogicEvaluator` in a terminal window.
2. execute `evalProp {insert program} {insert query}` where `program` can be self-made or chosen from the test cases provided in the code. Eg. `evalProp testProgram [(C1,C2)]`. (Remember that the `Atom` names are predefined in the top of the code as `data Atom`)

Level 2

Level 2 is implemented in `LogicEvaluator2.hs`. Level 2 is far more complex. The program follows the following ideas:

1. Determine if the query contains a variable. The result of the program is a `Bool` if there is no variable in the query and the result is a list of `Substitution`.
2. Expand the given query with the rules in the program. Expanding is

halted when a given predicate has only constant derivatives, so it preserves the variable names. The result will be a list of queries, since some predicates have more than one rule and the program has the examine all rules.
3. Unify will give all substitutions for every predicate.
4. All unifications will be intersected for each variable in the expanded query. The result of this technique is a list of all substitutions that will hold for each predicate in the query.
5. Depending on the resulting type, the program will either: Give back all substitutions or check if all constants in the query hold.

Usage :

1. Run `ghci LogicEvaluator2` in a terminal window.
2. Execute `evalOne {insert program} {insert query}` where `program` and `query` can be self-made or chosen from the test cases provided in the code. Eg. `evalOne testProgram query9`. (Remember that the `Predicate` names are predefined in the top of the code as `data Pred`)

Level 3

Level 3 is not implemented.