Logic-Based Application Example : Financial Advisor

As a final example, let's use FOL to represent and reason about a sample problem domain: financial analysis. Although simple, it illustrates many issues involved in real applications.

The function is to advise a user about whether to invest in stocks or savings. Some investors might want to split their money. Let's say that we have determined the following rules:

1. Individuals with inadequate savings should make increasing savings their top priority, regardless of income.
2. Individuals with adequate savings and adequate income should consider stocks, which are riskier but potentially more profit.
3. Individuals with a lower income and already have adequate savings should split their income between savings and stocks.

Our rule is to have at least $5000 in the bank for each dependent. An adequate income must be steady and supply at least $15000 per year plus $4000 for each dependent.

Let's start coding up some rules:

1. Savings_account(inadequate) $\rightarrow$ Investment(savings).
2. Savings_account(adequate) ^ Income(adequate) $\rightarrow$ Investment(stocks).
3. Savings_account(adequate) ^ Income(inadequate) $\rightarrow$ Investment(combination).

Now we have to determine when savings and income are adequate or inadequate. Let's define minsavings and the adequacy of savings:

MinSavings(Z) = 5000*Z

4. $\forall x$ Amount_saved(x) ^ $\exists y$ (dependents(y) ^ greater(x,MinSavings(y))) $\rightarrow$ Savings_Account(adequate)
5. $\forall x$ Amount_saved(x) ^ $\exists y$ (dependents(y) ^ not(greater(x,MinSavings(y)))) $\rightarrow$ Savings_Account(inadequate)

5a. $\neg\exists y$ dependents(y) $\rightarrow$ Savings_Account(adequate)

Now we need to define adequacy of income:

MinIncome(Z) = 15000 + (4000*Z)

6. $\forall x$ Earnings(x, steady) ^ $\exists y$ (dependents(y) ^ greater(x,MinIncome(y))) $\rightarrow$ income(adequate).
7. $\forall x$ Earnings(x, steady) ^ $\exists y$ (dependents(y) ^ Not(greater(x,MinIncome(y)))) $\rightarrow$ income(inadequate).
8. $\forall x$ Earnings(x,unsteady) $\rightarrow$ income(inadequate).

8a. $\forall x$ Earnings(x,steady) ^ $\neg\exists y$ dependents(y) $\rightarrow$ income(adequate)

That's all the knowledge we need for our system. Now let's give our system some sample data for Herman. Toiling as an instructor, Herman makes only $25000 a year. However, he had the good fortune of inheriting some money and has $22000 saved. He has three dependents.

9. amount_saved(22000)
10. earnings(25000, steady)
11. dependents(3)

Let's try to infer an investment strategy for Herman. We can use forward chaining and plug 10 and 11 into the first two components of premise 7.

Earnings(25000, steady) ^ dependents(3)

Unifies with

Earnings(X, steady) ^ dependents(y).

So this means that 25000=X and 3=Y.

Evaluating the function MinIncome(3) yields 15000+12000= 27000. So Not(greater(25000,27000)) is true, and via Modus Ponens we can now infer the RHS of rule 7: 12) income(inadequate).
We can also unify with assertion 4:

Amount_Saved(22000) ^ dependents(3) with 22000=X, 3=Y

Evaluating the function MinSavings(3) yields 15000.

We have Greater(22000,15000) so via Modus Ponens we can now infer the RHS of #4, or 13) savings_account(adequate).

With 12 and 13 we can now fire off rule #3 which suggests Investment(combination).

This is a simple example; more complex rules and knowledge is necessary for really useful advice. You might notice that there is no "magic" in all of this logic; it has essentially been a bunch of if-then-else statements combined with functions. With a large collection of rules, a program can appear magical and mystical but underneath the hood it's still just evaluating a bunch of rules. However some systems may exhibit some "emergent" characteristics, as a collection of rules in combination produce more intelligent behavior than any individual rule.