# Otter Theorem Proving Exercise

Brandon Bennett

October 26, 2007

This is an exercise in translation into 1st-order logic and basic operation of the Otter theorem prover.

## Using Otter

The Otter Linux executable can be found at `~krr/otter`.

Otter works on an input file containing a set of formulae and also instructions about what proof rules to use. Using these rules it trues to find a proof that the formula set is inconsistent. Thus to demonstrate that an argument of the form $\Gamma \vdash \phi$ is valid you must give otter the formula set $\Gamma \cup \{\neg\phi\}$ — if this is inconsistent the argument is valid.

## Example

Here is an example of an Otter file:

```
%               Schubert's "Steamroller" Problem
%
% Wolves, foxes, birds, caterpillars, and snails are animals,
% and there are some of each of them.
%
% Also there are some grains, and grains are plants.
%
% Every animal either likes to eat all plants or all animals much
% smaller than itself that like to eat some plants.
%
% Caterpillars and snails are much smaller than birds, which are much
% smaller than foxes, which are in turn much smaller than wolves.
%
% Wolves do not like to eat foxes or grains, while birds like to eat
% caterpillars but not snails.
%
% Caterpillars and snails like to eat some plants.
%
% Prove there is an animal that likes to eat a grain-eating animal.
%

set(auto).

formula_list(usable).

all x (Wolf(x) -> animal(x)).
all x (Fox(x) -> animal(x)).
all x (Bird(x) -> animal(x)).
all x (Caterpillar(x) -> animal(x)).
all x (Snail(x) -> animal(x)).
all x (Grain(x) -> plant(x)).
```

```
exists x Wolf(x).
exists x Fox(x).
exists x Bird(x).
exists x Caterpillar(x).
exists x Snail(x).
exists x Grain(x).

% All animals either eat all plants or eat all smaller animals that eat some plants.

all x (animal(x) -> (all y (plant(y)->eats(x,y)))
                    |
                    (all z ( animal(z) &
                             Smaller(z,x) &
                             (exists u (plant(u)&eats(z,u)))
                            ->
                             eats(x,z)))).

all x all y (Caterpillar(x) & Bird(y) -> Smaller(x,y)).
all x all y (Snail(x) & Bird(y) -> Smaller(x,y)).
all x all y (Bird(x) & Fox(y) -> Smaller(x,y)).
all x all y (Fox(x) & Wolf(y) -> Smaller(x,y)).
all x all y (Bird(x) & Caterpillar(y) -> eats(x,y)).

all x (Caterpillar(x) -> (exists y (plant(y) & eats(x,y)))).
all x (Snail(x)       -> (exists y (plant(y) & eats(x,y)))).

all x all y (Wolf(x) & Fox(y) -> -eats(x,y)).
all x all y (Wolf(x) & Grain(y) -> -eats(x,y)).
all x all y (Bird(x) & Snail(y) -> -eats(x,y)).

% negation of "there is an animal that eats {an animal that eats all grains}".
% Note the answer literal, which records the predator and the prey.

-(exists x exists y ( -$answer(eats(x,y)) &
      animal(x) &
      animal(y) &
      eats(x,y) &
              (all z (Grain(z) -> eats(y,z))))).
end_of_list.
```

The declaration `set(auto).` instructs Otter to choose its own inference rules (it usually makes a pretty good choice). '`formula_list(usable).`' just tells Otter that the next bit (up to '`end_of_list.`') is a list of formulae whose consistency is to be tested.

To run Otter just type:

```
~krr/otter < input.file
```

You may also want to redirect the output to a file:

```
~krr/otter < input.file > output.file
```

## The Exercise: A Murder Mystery Problem

Translate the following sentences into 1st-order logic and put them in a file using Otter's syntax.

1. Someone who lives in Dreadbury Mansion killed Aunt Agatha.
2. Agatha, the butler, and Charles live in Dreadbury Mansion, and are the only people who live therein.
3. A killer always hates his victim, and is never richer than his victim.
4. Charles hates no one that Aunt Agatha hates.
5. Agatha hates everyone except the butler.
6. The butler hates everyone not richer than Aunt Agatha.
7. The butler hates everyone Aunt Agatha hates.
8. No one hates everyone.
9. Agatha is not the butler.

Now use the Otter theorem prover to deduce who killed Aunt Agatha.