

An Brief Incomplete Introduction to Otter

This introduction simplifies many aspects of Otter, focusing on one way of using Otter to perform logical inference. Much more information can be found in the Otter manual, which is in `/home/bylander/otter/documents/otter33.pdf`, and the official Otter web site <http://www-unix.mcs.anl.gov/AR/otter/>, where you can download the code.

Otter (Organized Techniques for Theorem-proving and Effective Research) is a resolution-style theorem prover written in C. Otter implements the resolution inference rule and unit resolution inference rule (respectively called “binary resolution” and “unit deletion” in Otter). Otter also implements other inference rules that we will not discuss.

To use Otter, one must specify the parameters of its inference procedure, the clauses to be refuted, and the clauses can be used in the proof. All of these are put in an input file. If `file.in` is the input file, one can then call Otter by:

```
setenv OTTER /home/bylander/otter/bin-SunOS/otter
or OTTER=/home/bylander/otter/bin-SunOS/otter
$OTTER < file.in
```

Parameters

The parameters of the inference procedure are specified at the beginning of the program. For this class, I suggest that you always use the following:

```
clear(print_given).
set(binary_res).
clear(detailed_history).
clear(print_kept).
clear(print_back_sub).
assign(max_seconds,1).
assign(stats_level,0).
```

Note that every line ends with a period. Commenting out any of the `clear` lines, as well as increasing the `stats_level`, will produce more output. A comment starts with a `%` and goes to the end of the line.

Propositional Logic Example

This example show how Otter performs the inference of $W_{1,3}$ in the wumpus world from the book, using only the resolution inference rule.

Usable List

The clauses to be used are put in the usable list. The first three lines list the perceptions. The next three lines represent implications from the lack of a stench. The last line represents the implication from a stench.

```

list(usable).
-S11.   -B11.
-S21.   B21.
S12.    -B12.
S11 | -W11.    S11 | -W12.    S11 | -W21.
S21 | -W11.    S21 | -W21.    S21 | -W22.    S21 | -W31.
S12 | -W11.    S12 | -W12.    S12 | -W22.    S12 | -W13.
-S12 | W11 | W12 | W22 | W13.
end_of_list.

```

The | operator corresponds to OR, and the - operator corresponds to negation. S11, B11, S21, B21, S12, B12, W11, W12, W21, W22, W31, and W13 are propositional variables. Note that each clause ends with a period.

Sos List

The statement to be proved is negated and put into the sos list.

```

list(sos).
-W13.
% -W13 | $answer.
end_of_list.

```

In this example, we want to prove that the wumpus is in square [1, 3]. In resolution theorem proving, this means that we want to refute that the wumpus is not in square [1, 3]. In other words, we want to prove that the wumpus is in [1, 3] using a proof by contradiction. The first step of such a proof is to assert the negation of what you want to prove. You might like it better to use the commented out line instead, which corresponds to saying an answer is implied if the wumpus is in [1, 3] (yes, you need to use \$answer; don't get creative here).

Result

When the parameters, the usable list, and the sos list are all put into a file that is input to Otter, then the output includes the following proof:

```

1 [] -S11.
3 [] -S21.
5 [] S12.
8 [] S11| -W12.
10 [] S21| -W11.
12 [] S21| -W22.
18 [] -S12|W11|W12|W22|W13.
19 [] -W13.
20 [binary,19,18,unit_del,5] W11|W12|W22.
21 [binary,20,10,unit_del,3] W12|W22.
24 [binary,21,8,unit_del,1] W22.
26 [binary,24,12] S21.
27 [binary,26,3] $F.

```

Each clause that is given or inferred is numbered. Some numbers are skipped because they correspond to other clauses that were given or inferred, but were not part of the proof. Lines 1 through 19 are clauses given in the usable list or the sos list. Line 20 shows two inferences: (binary) resolution was applied to clauses 19 and 18, followed by unit resolution/deletion with clause 5 to infer that the wumpus is in [1, 1], [1, 2], or [2, 2]. This is something that would be true from the stench [1, 2] (clause 5) and if it was the case that the wumpus is not in [1, 3] (clause 19). The sequence of inferences proceeds to line 27, where false (**\$F**) is inferred, i.e., a contradiction is inferred. Because asserting that the wumpus is not in [1, 3] led to a contradiction, we can conclude that the wumpus is in [1, 3].

Predicate Logic Example

This example infers an ancestor relationship from knowledge about who is a mother/father of whom.

Usable List

The clauses to be used are put in the usable list.

```
list(usable).
mother(Liz,Charley).
father(Charley,Billy).
-mother(x,y) | parent(x,y).
-father(x,y) | parent(x,y).
-parent(x,y) | ancestor(x,y).
-parent(x,y) | -ancestor(y,z) | ancestor(x,z).
end_of_list.
```

The `|` operator corresponds to OR, and the `-` operator corresponds to negation. `mother`, and `father`, `parent`, and `ancestor`, are predicates. `x` and `y` are variables. `Liz`, `Charley`, and `Billy` are constants. The way that Otter distinguishes variables from constants is that variables start with a lower-case `u`, `v`, `w`, `x`, `y`, or `z`.

The six formulas are intended to have the following meanings:

Liz is the mother of Charley.
 Charley is the father of Billy,
 If x is the mother of y , then x is a parent of y .
 If x is the father of y , then x is a parent of y .
 If x is a parent of y , then x is an ancestor of y .
 If x is a parent of y and y is an ancestor of z , then x is an ancestor of z .

These statements can be represented in first-order predicate logic by:

$$\begin{aligned} &\forall x, y (parent(x, y) \rightarrow ancestor(x, y)) \\ &\forall x, y ((parent(x, y) \wedge ancestor(y, z)) \rightarrow ancestor(x, z)) \\ &\forall x, y (mother(x, y) \rightarrow parent(x, y)) \\ &\forall x, y (father(x, y) \rightarrow parent(x, y)) \end{aligned}$$

```
mother(Liz, Charley)
father(Charley, Billy)
```

The first-order statements are converted to clause form by removing universal quantifiers, and transforming to disjunctions.

Sos List

The statement to be proved is negated and put in the sos list.

```
list(sos).
-ancestor(Liz, Billy).
% -ancestor(Liz, Billy) | $answer.
end_of_list.
```

In this example, we want to prove that Liz is an ancestor of Billy. In resolution theorem proving, this means that we want to refute that Liz is not an ancestor of Billy. In other words, we want to prove that Liz is an ancestor of Billy using a proof by contradiction. The first step of such a proof is to assert the negation of what you want to prove. You might like it better to use the commented out line instead, which corresponds to saying an answer is implied if Liz is an ancestor of Billy (yes, you need to use `$answer`; don't get creative here).

Result

When the parameters, the usable list, and the sos list are all put into a file that is input to Otter, then the output includes the following proof:

```
1 [] -parent(x,y) | ancestor(x,y).
2 [] -parent(x,y) | -ancestor(y,z) | ancestor(x,z).
3 [] -mother(x,y) | parent(x,y).
4 [] -father(x,y) | parent(x,y).
5 [] mother(Liz,Charley).
6 [] father(Charley,Billy).
7 [] -ancestor(Liz,Billy).
8 [binary,7,2] -parent(Liz,x) | -ancestor(x,Billy).
13 [binary,8,3] -ancestor(x,Billy) | -mother(Liz,x).
20 [binary,13,5] -ancestor(Charley,Billy).
22 [binary,20,1] -parent(Charley,Billy).
23 [binary,22,4] -father(Charley,Billy).
24 [binary,23,6] $F.
```

Each clause that is given or inferred is numbered. Some numbers are skipped because they correspond to other clauses that were given or inferred, but were not part of the proof. Lines 1–7 are clauses given in the usable list or the sos list. Line 8 shows that (binary) resolution was applied to clauses 7 and 2 to infer that, for all x , Liz is not a parent of x , or x is not an ancestor of Billy. This is something that would be true if it was the case that Liz is not an ancestor of Billy (clause 7). The sequence of inferences proceeds to line 24, where false ($\$F$) is inferred, i.e., a contradiction is inferred. Because asserting Liz is not an ancestor of Billy led to a contradiction, we can conclude that Liz is an ancestor of Billy.