Programming Languages

Prolog Part 1
Propositions:
Statements that can be either True or False

Truth:  
Are there well formed propositional formulas (i.e., Statements) that return True when their input is True

\[
\text{truth1} :: (\text{Bool} \rightarrow \text{Bool}) \rightarrow \text{Bool} \\
\text{truth1 wff} = (\text{wff True})
\]

\[
\text{truth2} :: (\text{Bool} \rightarrow \text{Bool} \rightarrow \text{Bool}) \rightarrow \text{Bool} \\
\text{truth2 wff} = (\text{wff True True})
\]

\[
(\ \lambda \ p \rightarrow \text{not} \ p) \\
(\ \lambda \ p \ q \rightarrow (\text{p \&\& \ q}) \big\| (\text{not} \ p \rightarrow \text{q})) \\
(\ \lambda \ p \ q \rightarrow \text{not} \ p \rightarrow \text{q}) \\
(\ \lambda \ p \ q \rightarrow (\text{not} \ p \&\& \ q) \&\& (\text{not} \ p \rightarrow \text{q}))
\]

<table>
<thead>
<tr>
<th>Clausal Form</th>
<th>Conventional Syntax</th>
<th>PROLOG</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ p }</td>
<td>p</td>
<td>p.</td>
</tr>
<tr>
<td>{ q }</td>
<td>q</td>
<td>q.</td>
</tr>
<tr>
<td>{ r }</td>
<td>r</td>
<td>r.</td>
</tr>
<tr>
<td>{ s, \neg p, \neg q }</td>
<td>p &amp;&amp; q \rightarrow s</td>
<td>s:-p,q.</td>
</tr>
<tr>
<td>{ t, \neg s, \neg r }</td>
<td>s &amp; r \rightarrow t</td>
<td>t:-s,r.</td>
</tr>
<tr>
<td>{ \neg t }</td>
<td>\neg t \rightarrow \bot</td>
<td>\neg t .</td>
</tr>
</tbody>
</table>

If it was never possible for it not to be True that something was going to exist, and it will never be possible for it not to be True that something existed in the past then it is impossible for Truth ever to have had a beginning or ever to have an end. That is, it was never possible that Truth cannot be conceived not to exist.

If R is something that can be conceived not to exist and T is something that cannot be conceived not to exist and T is greater than R and God is that, than which nothing greater can be conceived, then God exists and is Truth.
How Robots Reason

How does a robot scientist “reason?” It uses the same options people use. One is deductive inference, which is the foundation for mathematics and computer science. Deductive reasoning is “sound.” That is, if you start with truth you can infer only new truths. Unfortunately, in the absence of a consummate “theory of everything,” deduction is insufficient for science, because it can work out only the consequences of what is already known.

A second option, abductive reasoning, is not sound, as is obvious from the swan example below; many things are white but are not swans. Yet abduction does provide a way of generating hypotheses that may be true. The great insight of science is that the way to decide truth is not by pure deduction from assumptions but rather by experimenting on the physical world. If Adam hypothesizes that Daisy is a swan, then the way to decide on the truth of this proposition is for Adam to experimentally catch Daisy and test whether she is a swan, a duck or something else.

Induction, like abduction, provides a way to infer new hypotheses. If every swan we see is white, it is natural to infer, as Aristotle actually did, that all swans are white. But induction is not sound, and Aristotle’s induction was disproved by the discovery of black swans in Australia. We constantly use induction in our daily lives. It reassures us that the sun will rise tomorrow and that our breakfast won’t poison us. Induction’s role in science is controversial, however, because the main justification for induction is that it generally works, which is itself an induction.

76 Scientific American, January 2011
Composition of Relations

Composing Relations. Suppose that \( R \) and \( S \) are relations on \( A \). The \textit{composition} \( R \circ S \) of \( R \) and \( S \) is the relation on \( A \) that is defined by

\[
x(R \circ S)z \equiv \exists y \in A(xRy \land ySz).
\]

Furthermore, for \( n \in \mathbb{N}, n \geq 1 \) we define \( R^n \) by means of \( R^1 := R \), \( R^{n+1} := R^n \circ R \).

\[
\begin{align*}
\text{Composing Relations} & \quad \{(1,2),(2,3),(2,4),(2,5),(2,6),(6,7),(6,8)\} \\
\text{ComposeR r 2} & \quad \{(1,3),(1,4),(1,5),(1,6),(2,7),(2,8)\} \quad \text{composed with} \\
\text{ComposeR r 3} & \quad \{(1,7),(1,8)\} \quad \text{composed with} \\
\text{ComposeR r 4} & \quad \{\} \quad \text{composed with}
\end{align*}
\]
Composition of Relations

Composing Relations. Suppose that $R$ and $S$ are relations on $A$. The composition $R \circ S$ of $R$ and $S$ is the relation on $A$ that is defined by

$$x(R \circ S)z \equiv \exists y \in A(xRy \wedge ySz).$$

Furthermore, for $n \in \mathbb{N}, n \geq 1$ we define $R^n$ by means of $R^1 := R$, $R^{n+1} := R^n \circ R$.

If 1 is rose, 2 is phil, 3 is nicolette, 4 is antoinette, 5 is jeanette, 6 is philJ, 7 is philJJ and 8 is patrick

<table>
<thead>
<tr>
<th>$r$</th>
<th>{(1,2),(2,3),(2,4),(2,5),(2,6),(6,7),(6,8)}</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ComposeR $r \ 2$</th>
<th>{(1,3),(1,4),(1,5),(1,6),(2,7),(2,8)}</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>ComposeR $r \ 3$</th>
<th>{(1,7),(1,8)}</th>
</tr>
</thead>
</table>

| ComposeR $r \ 4$ | \{} |

**Great Grandparent Relation**

**Grandparent Relation**

{(rose,phil),(phil,nicolette),(phil,antoinette),(phil,jeanette),
(phil,philJ),(philJ,philJJ),(philJ,patrick)}

{(rose,nicolette),(rose,antoinette),(rose,jeanette),(rose,philJ),
(phil,philJJ),(phil,patrick)}
Prolog Example

parent(hank,ben).
parent(hank,denise).
parent(irene,ben).
parent(irene,denise).
parent(alice,carl).
parent(ben,carl).
parent(denise,frank).
parent(denise,gary).
parent(earl,frank).
parent(earl,gary).
grandparent(X,Z) :- parent(X,Y), parent(Y,Z).

Besides being a Prolog Program this is also an example of a Compression.
Composition of Relations

This is not the last time we’ll see something similar to Composition of Relations:

\[
\text{grandparent}(X,Z) :- \text{parent}(X,Y), \text{parent}(Y,Z). \\
\]

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{EMPNO} & \text{ENAME} & \text{JOB} & \text{SAL} & \text{DNAME} \\
\hline
1 & 7782 & CLARK & MANAGER & 2450 \text{ ACCOUNTING} \\
2 & 7934 & MILLER & CLERK & 1300 \text{ ACCOUNTING} \\
3 & 7833 & KING & PRESIDENT & 5000 \text{ ACCOUNTING} \\
4 & 7902 & FORD & ANALYST & 3000 \text{ RESEARCH} \\
5 & 7788 & SCOTT & ANALYST & 3000 \text{ RESEARCH} \\
6 & 7566 & JONES & MANAGER & 2975 \text{ RESEARCH} \\
7 & 7369 & SMITH & CLERK & 800 \text{ RESEARCH} \\
8 & 7875 & ADAMS & CLERK & 1100 \text{ RESEARCH} \\
9 & 7521 & WARD & SALESMAN & 1250 \text{ SALES} \\
10 & 7654 & MARTIN & SALESMAN & 1250 \text{ SALES} \\
11 & 7844 & TURNER & SALESMAN & 1600 \text{ SALES} \\
12 & 7900 & JAMES & CLERK & 950 \text{ SALES} \\
13 & 7499 & ALLEN & SALESMAN & 1600 \text{ SALES} \\
14 & 7698 & BLAKE & MANAGER & 2650 \text{ SALES} \\
\hline
\end{array}
\]

List Comprehension

\[
\text{Main}> \{ (\text{empno}, \text{ename}, \text{job}, \text{sal}, \text{dname}) | \ (\text{empno}, \text{ename}, \text{job}, _, _, \text{sal}, \text{edepn}) \leftarrow \text{emp}, (\text{deptno, dname, loc}) \leftarrow \text{dept, } \text{edepn} \leftarrow \text{deptno} \}
\]
Running Prolog

To download prolog for windows go to: http://www.gprolog.org/#download

For Linux it should be - /usr/opt/gprolog-1.3.0/bin/gprolog

You need to set your PATH environment variable as follows:
export PATH="/usr/opt/gprolog-1.3.0/bin":$PATH

Make sure you’re using the “bash” shell for this.

Don’t forget the dot

Turn on trace.

Turn off trace.
Gödel's Incompleteness Theorem

If “proof” is a proof of “statement” then P is True.

If you have a statement g with variable x and if when you substitute g for x and you produce “statement” then Q is True.

\[
\text{not } P(\text{proof, statement}) \land \land Q(x, \text{statement}) = g
\]

Let g be the Gödel number for this statement,

A recursive notion.

\[
\text{not } P(\text{proof, statement}) \land \land Q(g, \text{statement}) = s
\]

Let s be the Gödel number for this statement but by the definition of Q that means “statement” is “s”.

\[
\text{not } P(\text{proof, s}) \land \land Q(g, s) - \text{I am a statement that is not provable.}
\]

→ There are Predicate Logic Statements that are True that can’t be proved True (Incompleteness) and/or there are Predicate Logic Statements that can be proved True that are actually False (→ Inconsistent Axioms or Unsound inference rules).

i.e., If Gödel's statement is true, then it is an example of something that is true for which there is no proof.

If Gödel's statement is false, then it has a proof and that proof proves the false Gödel statement true.
Gödel's Incompleteness Theorem
## Limitive Theorems

<table>
<thead>
<tr>
<th>Theorem</th>
<th>Language of Psychology</th>
<th>Language of Physics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gödel's first incompleteness theorem</td>
<td>There is no consistent human computer capable of formulating a program which, if carried out, would produce all the true and only the true sentences of arithmetic.</td>
<td>There is no consistent computing machine which can be programmed to produce all the true and only the true sentences of arithmetic.</td>
</tr>
<tr>
<td>Gödel's second incompleteness theorem</td>
<td>No consistent human computer can prove an effective or constructive expression of its own consistency (see page 222). (This depends on Church’s theorem).</td>
<td>No consistent computing machine can be programmed to prove a computable or computably enumerable expression of its own consistency (see page 222). (This depends on Church’s theorem).</td>
</tr>
<tr>
<td>Church’s theorem</td>
<td>There exists a set of problems which no consistent human computer can solve.</td>
<td>There exists a set of problems which no consistent computing machine can be programmed to solve.</td>
</tr>
<tr>
<td>Skolem’s theorem</td>
<td>No consistent human computer can (categorically) formalize the notion of natural number.</td>
<td>No consistent computing machine can be programmed to produce a (categorical) formalization of the motion of natural number.</td>
</tr>
</tbody>
</table>
Horn Clause

parent(hank,ben).
parent(hank,denise).
parent(irene,ben).
parent(irene,denise).
parent(alice,carl).
parent(ben,carl).
parent(denise,frank).
parent(denise,gary).
parent(earl,frank).
parent(earl,gary).
grandparent(X,Z) :- parent(X,Y) , parent(Y,Z).

\[ \logEquiv2 (\neg p \land q \rightarrow p \implies q) (\land p q \rightarrow \neg p \lor q) \]
\[ \neg(\neg parent(X,Y), parent(Y,Z)) \lor \neg grandparent(X,Z) \]
\[ \logEquiv2 (\neg p \land q \rightarrow \neg (p \land \neg q)) (\land p q \rightarrow \neg p \lor \neg q) \]
\[ \neg(\neg parent(X,Y)) \lor \neg(\neg parent(Y,Z)) \lor \neg grandparent(X,Z) \]

A Horn clause is a disjunction of Predicates in which at most one of the Predicates is not negative
Horn Clause?

reads(X) || writes(X) :- literate(X).

not(literate(X)) || reads(X) || writes(X)

logEquiv2 (\ p q -> p ==> q) (\ p q -> not p || q)

Prolog only deals with Horn Clauses
Proof by Contradiction

1. Let \( P = \text{It’s raining, I’m outside} \)
2. \( P_1 \). (\( P_1 \) is True, i.e., it’s raining)
3. \( P_2 \). (\( P_2 \) is True, i.e., I’m outside)
4. \( Q : - P = \text{I’m wet} : - \text{It’s raining, I’m outside}. \) (if it’s raining and I’m outside then I’m wet)
5. \( -Q \) (To answer the Query “Am I wet” against the Database, assume I’m not wet)
6. \( -\text{(It’s raining, I’m outside)} \) (From 4 and 5 and Pattern 1)
7. \( -\text{I’m outside} \) (From 2 and 6 and Pattern 2)
8. Contradiction – Therefore I’m wet (From 3 and 7 and Pattern 3)

It is now my intention to follow another and, as I think, a very beautiful way of proving these same truths without the help of any assumption. I shall proceed as follows: I shall take the contradictory of the proposition to be proved and elicit the required result from this by a straight-forward demonstration – Gerolamo Sachere (1167-1733).
Proof by Contradiction, Unification, Resolution and Backtracking

Pattern 1 (Modus Tollens):
Q :- (P1, P2).
-Q
⇒ -(P1, P2)

Pattern 2 (Affirming a Conjunct):
P1.
-(P1, P2)
⇒ -P2

Pattern 3:
P2.
-P2
⇒ Contradiction

1). parent(hank, ben).
2). parent(ben, carl).
3). parent(ben, sue).
4). grandparent(X, Z) :- parent(X, Y), parent(Y, Z).
5). –grandparent(A, B)
   (Unify A to X) (Unify B to Z) then Resolve 5 & 4
6). –(parent(A, Y), parent(Y, B)).
   (Unify A to hank) (Unify Y to ben)
   (Unify B to carl) then Resolve 6 & 1
7). –parent(ben, carl)
Contradiction ⇒ grandparent(hank, carl)
Backtrack to 6 and
   (Unify B to sue) then Resolve 6 & 1
9). –parent(ben, sue)
Contradiction ⇒ grandparent(hank, sue)
Proof by Contradiction, Unification, Resolution and Backtracking

1). parent(hank,ben).
2). parent(ben,carl).
3). parent(ben,sue).
4). grandparent(X,Z) :- parent(X,Y), parent(Y,Z).
5). ~grandparent(A, B)
   (Unify A to X) (Unify B to Z) then Resolve 5 & 4
6). ~parent(A, Y), parent(Y, B).
   (Unify A to hank) (Unify Y to ben)
   (Unify B to carl) then Resolve 6 & 1
7). ~parent(ben, carl)
   Contradiction \( \rightarrow \) grandparent(hank, carl)
   Backtrack
   (Unify B to sue) then Resolve 6 & 1
9. ~parent(ben, sue)
   Contradiction \( \rightarrow \) grandparent(hank, sue)

Dr. Philip Cannata
Some Sample Prolog Programs

factorial(0, 1).
factorial(N, A) :- N > 0, M is N - 1, factorial(M, A1), A is N * A1.

appendList([], L, L).
appendList([X|Y], L, [X|A]) :- appendList(Y, L, A).

mapsq([], []).
mapsq([X|Y], [Z|A]) :- Z is X*X, mapsq(Y, A).

even([], []).
even([X|Y], [X|A]) :- 0 is X mod 2, even(Y, A).
even([X|Y], A) :- 1 is X mod 2, even(Y, A).

map(FunctionName, [H|T], [NH|NT]) :- Function=..[FunctionName,H,NH], call(Function), map(FunctionName, T, NT).
map(_, [], []).
neg(A, B) :- B is -A.
inc(A, B) :- B is A+1.
dec(A, B) :- B is A-1.

lambda(P, Body, R) :-
    Function=..[Body, P, R],
call(Function).

% Remember(let (x 5) x*2) -> (lambda x: x*2)(5)
let(V, Body, A) :- lambda(V, Body, A).
Proof by Contradiction, Unification, Resolution and Backtracking

1). factorial(0, 1).
2). factorial(N, Result) :- N > 0, M is N -1, factorial(M, S), Result is N * S.
3). –factorial(2, X)
   (Unify 2 to N) (Unify X to Result) then Resolve 3 & 2
6). –(2 > 0, M is 1, factorial(1, S), X is 2 * S.
   (Unify 1 to N) (Unify S to Result) then Resolve 6 & 2
7). –(1 > 0, M is 0, factorial(0, S1), S is 1 * S1.
   (Unify 0 to N) (Unify S1 to Result) then Resolve 7 & 2
9). –factorial(0, 1)
Contradiction  →  There is a factorial 2, now return from the proof with the answer.

Pattern 1 (Modus Tollens):
Q :- (P1, P2).
- Q
→ -(P1, P2)

Pattern 2 (Affirming a Conjunct):
P1.
-(P1, P2)
→ -P2

Pattern 3: P2.
-P2
→ Contradiction
### Hmm Runtime Stack for Factorial 3

```c
int factorial(int n) {
    if(n < 1) {
        return 1;
    } else {
        return n * factorial(n - 1);
    }
}

int main() {
    int number, answer;
    number = 3;
    answer = factorial(number);
    print(answer);
}
```
Proof by Contradiction, Unification, Resolution and Backtracking

1). factorial(0, 1).
2). factorial(N, Result) :- N > 0, M is N -1, factorial(M, S), Result is N * S.
3). –factorial(2, _16)
   (Unify 2 to N) (Unify _16 to Result) then Resolve 3 & 2
6). –(2 > 0, _113 is 1, factorial(1, _138), _16 is 2 * _138.
   (Unify 1 to N) (Unify _138 to Result) then Resolve 6 & 2
7). –(1 > 0, _190 is 0, factorial(0, S_215), _138 is 1 * _215.
   (Unify 0 to N) (Unify S_215 to Result) then Resolve 7 & 2
9). –factorial(0, 1)
   Contradiction → There is a factorial of 2, now return from the proof with the answer.

Note: the trace shows _243 is 1*1 but then that value gets moved into _138
Building Problem part 1

Baker, Cooper, Fletcher, Miller and Smith live in a five-story building.

Pattern 1 (Modus Tollens):

\[ Q :- (P1, P2). \]
\[ \neg Q \Rightarrow \neg (P1, P2) \]

Pattern 2 (Affirming a Conjunct):

\[ P1. \neg (P1, P2) \Rightarrow \neg P2 \]

Pattern 3:

\[ P2. \neg P2 \Rightarrow \text{Contradiction} \]

```prolog
floors([floor(_,5), floor(_,4), floor(_,3), floor(_,2), floor(_,1)]).
building(Floors) :- floors(Floors),
    bmember(floor(baker, B), Floors),
    bmember(floor(cooper, C), Floors),
    bmember(floor(fletcher, F), Floors),
    bmember(floor(miller, M), Floors),
    bmember(floor(smith, S), Floors).

bmember(X, [X | _]).

bmember(X, [_ | Y]) :- bmember(X, Y).

?-building(X)
```

This Prolog Code can be found in 11Prolog Examples.p
Building Problem part 2
Baker, Cooper, Fletcher, Miller and Smith live in a five-story building. Baker doesn’t live on the 5th floor and Cooper doesn’t live on the 1st floor. Fletcher doesn’t live on the top or bottom floors, and he is not on a floor adjacent to Smith or Cooper. Miller lives on the some floor above Cooper. Who lies on what floors?

<table>
<thead>
<tr>
<th>Pattern 1 (Modus Tollens):</th>
<th>floors([floor(<em>,5),floor(</em>,4),floor(<em>,3),floor(</em>,2),floor(_,1)]).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q :- (P1, P2).</td>
<td>building(Floors) :- floors(Floors),</td>
</tr>
<tr>
<td>-Q</td>
<td>bmemeber(floor(baker, B), Floors), B = 5,</td>
</tr>
<tr>
<td>\rightarrow -(P1, P2)</td>
<td>bmemeber(floor(cooper, C), Floors), C = 1,</td>
</tr>
<tr>
<td></td>
<td>bmemeber(floor(fletcher, F), Floors), F = 1, F = 5,</td>
</tr>
<tr>
<td></td>
<td>bmemeber(floor(miller, M), Floors), M &gt; C,</td>
</tr>
<tr>
<td></td>
<td>bmemeber(floor(smith, S), Floors), not(adjacent(S, F)),</td>
</tr>
<tr>
<td></td>
<td>+ adjacent(F, C),</td>
</tr>
<tr>
<td></td>
<td>print_floors(Floors).</td>
</tr>
<tr>
<td></td>
<td>print_floors([A</td>
</tr>
<tr>
<td></td>
<td>print_floors([]).</td>
</tr>
<tr>
<td></td>
<td>bmemeber(X, [X</td>
</tr>
<tr>
<td></td>
<td>bmemeber(X, [_</td>
</tr>
<tr>
<td></td>
<td>adjacent(X, Y) :- X =:= Y+1.</td>
</tr>
<tr>
<td></td>
<td>adjacent(X, Y) :- X =:= Y-1.</td>
</tr>
<tr>
<td></td>
<td>not(Goal) :- + call(Goal).</td>
</tr>
<tr>
<td>Pattern 2 (Affirming a Conjunct):</td>
<td></td>
</tr>
<tr>
<td>P1.</td>
<td></td>
</tr>
<tr>
<td>-(P1, P2)</td>
<td></td>
</tr>
<tr>
<td>\rightarrow -P2</td>
<td></td>
</tr>
<tr>
<td>Pattern 3:</td>
<td></td>
</tr>
<tr>
<td>P2.</td>
<td></td>
</tr>
<tr>
<td>-P2</td>
<td></td>
</tr>
<tr>
<td>\rightarrow Contradiction</td>
<td></td>
</tr>
</tbody>
</table>

This Prolog Code can be found in 11PrologBuilding.p
n Queens Problem in Haskell

The **n queens puzzle** is the problem of placing *n* chess queens on an *n*×*n* chessboard such that none of them are able to capture any other using the standard chess queen's moves. Here's a Haskell solution:

```haskell
queen n = solve n
    where
    solve 0 = [[]]
solve (k+1) = [q:b | b <- solve k,
                    q <- [0..(n-1)],
                    safe q b ]

safe q b = and [not (checks q b i) |
                i <- [0..(length b - 1)] ]

checks q b i = q == b!!i || abs(q - b!!i) == i+1
```

The following code might be easier to think about first:

```haskell
queen n = solve n
    where
    solve 0 = [[]]
solve (k+1) = [q:b | b <- solve k, q <- [0..(n-1)]]
```
n Queens Problem in Haskell

```
queen n = solve n
  where
  solve 0 = [[]]
  solve (k+1) = [q:b | b <- solve k, q <- [0..(n-1)] ]
```

Put this code into initQueens.h

Hugs> :l initQueens.h
Main> queen 4
[[[0,0,0],[1,0,0],[2,0,0],[3,0,0]],
 [[0,1,0],[1,1,0],[2,1,0],[3,1,0]],
 [[0,2,0],[1,2,0],[2,2,0],[3,2,0]],
 [[0,3,0],[1,3,0],[2,3,0],[3,3,0]]]

Dr. Philip Cannata
**n Queens Problem in Prolog**

The n queens puzzle is the problem of placing n chess queens on an n×n chessboard such that none of them are able to capture any other using the standard chess queen's moves. Here’s a Prolog solution:

```
/* valid(TrialRow, TrialSwDiag, TrialSeDiag, RowList, SwDiagList, SeDiagList) */
valid(_, _, _, []).
valid(TrialRow, TrialSwDiag, TrialSeDiag, RowList, SwDiagList, SeDiagList) :-
    not(member(TrialRow, RowList)),
    not(member(TrialSwDiag, SwDiagList)),
    not(member(TrialSeDiag, SeDiagList)).

place(N, Row, Col, RowList, SwDiagList, SeDiagList, Answer) :-
    Row < N,
    getDiag(Row, Col, SeDiag, SwDiag),
    valid(Row, SeDiag, SwDiag, RowList, SwDiagList, SeDiagList).

place(N, Row, Col, RowList, SwDiagList, SeDiagList, Answer) :-
    NextRow is Row + 1,
    NextRow < N,
    place(N, NextRow, Col, RowList, SwDiagList, SeDiagList, Answer).

/* compute SeDiag, SwDiag */
getDiag(Row, Col, SwDiag, SeDiag) :-
    SwDiag is Row + Col, SeDiag is Row - Col.

solve(N, Col, RowList, SwDiagList, SeDiagList, Answer) :-
    Col >= N.
solve(N, Col, RowList, SwDiagList, SeDiagList, Answer) :-
    Col < N,
    place(N, 0, Col, RowList, SwDiagList, SeDiagList, Answer),
    getDiag(Row, Col, SwDiag, SeDiag),
    NextCol is Col + 1,
    solve(N, NextCol, [Row | RowList], [SwDiag | SwDiagList],
          [SeDiag | SeDiagList], Answer).
queens(N, Answer) :- solve(N, 0, [ ], [ ], [ ], Answer).
```

```
/* iterate over columns, placing queens */
solve(N, Col, RowList, _, _, RowList) :- Col >= N.
solve(N, Col, RowList, SwDiagList, SeDiagList, Answer) :-
    Col < N,
    place(N, 0, Col, RowList, SwDiagList, SeDiagList, Answer),
    getDiag(Row, Col, SwDiag, SeDiag),
    NextCol is Col + 1,
    solve(N, NextCol, [Row | RowList], [SwDiag | SwDiagList],
          [SeDiag | SeDiagList], Answer).
queens(N, Answer) :- solve(N, 0, [ ], [ ], [ ], Answer).
```
n Queens Problem in Prolog

This might be easier to understand initially: | ?- queens(3,R).

place(N, Row, Col, RowList, Row) :-
  Row < N.

place(N, Row, Col, RowList, Answer) :-
  NextRow is Row + 1,
  NextRow < N,
  place(N, NextRow, Col, RowList, Answer).

/* iterate over columns, placing queens */
solve(N, Col, RowList, RowList) :- Col >= N.

solve(N, Col, RowList, Answer) :-
  Col < N,
  place(N, 0, Col, RowList, Row),
  NextCol is Col + 1,
  solve(N, NextCol, [Row | RowList], Answer).

queens(N, Answer) :- solve(N, 0, [ ], Answer).

place(N, Row, Col, RowList, Row) :-
  Row < N.

place(N, Row, Col, RowList, Answer) :-
  NextRow is Row + 1,
  NextRow < N,
  place(N, NextRow, Col, RowList, Answer).

/* iterate over columns, placing queens */
solve(N, Col, RowList, RowList) :- Col >= N.

solve(N, Col, RowList, Answer) :-
  Col < N,
  place(N, 0, Col, RowList, Row),
  NextCol is Col + 1,
  solve(N, NextCol, [Row | RowList], Answer).

queens(N, Answer) :- solve(N, 0, [ ], Answer).

/* valid(TrialRow, TrialSwDiag, TrialSeDiag,
   RowList, SwDiagList, SeDiagList) */
valid(_, _, _, [ ]).
valid(TrialRow, TrialSwDiag, TrialSeDiag,
   RowList, SwDiagList, SeDiagList) :-
  not(member(TrialRow, RowList)),
  not(member(TrialSwDiag, SwDiagList)),
  not(member(TrialSeDiag, SeDiagList)).

bmember(X, [X | _]).
bmember(X, [_ | Y]) :- bmember(X, Y).
not(Goal) :- \+ call(Goal).

/* compute SeDiag, SwDiag */
getDiag(Row, Col, SwDiag, SeDiag) :-
  SwDiag is Row + Col, SeDiag is Row - Col.
Lewis Carroll Logic Paradox

Unlike others (http://www.paradoxhumanity.com/2011/06/the-barbershop-paradox/), I think this was written as a satire on Reductio ad Absurdum.

A Logical Paradox

“What, nothing to do?” said Uncle Jim. “Then come along with me down to Allen’s. And you can just take a turn while I get myself shaved.”

“All right,” said Uncle Joe. “And the Cub had better come too, I suppose?”

The “Cub” was me, as the reader will perhaps have guessed for himself. I’m turned fifteen—more than three months ago; but there’s no sort of use in mentioning that to Uncle Joe; he’d only say “go to your cubbicle, little boy!” or “Then I suppose you can do cubic equations?” or some equally vile pun. He asked me yesterday to give him an instance of a Proposition in A. And I said “All uncles make vile puns”. And I don’t think he liked it. However, that’s neither here nor there. I was glad enough to go. I do love hearing those uncles of mine “chop logic,” as they call it; and they’re desperate hands at it, I can tell you!

“That is not a logical inference from my remark,” said Uncle Jim.

“Never said it was,” said Uncle Joe: “it’s a Reductio ad Absurdum”.


That’s the sort of way they always go on, whenever I’m with them. As if there was any fun in calling me a Minor!

After a bit, Uncle Jim began again, just as we came in sight of the barber’s. “I only hope Carr will be at home,” he said. “Brown’s so clumsy. And Allen’s hand has been shaky ever since he had that fever.”

“Carr’s certain to be in,” said Uncle Joe.

“I’ll bet you sixpence he isn’t!” said I.

“Keep your bets for your betters,” said Uncle Joe. “I mean”—he hurried on, seeing by the grin on my face what a slip he’d made—“I mean that I can prove it, logically. It isn’t a matter of chance.”

“Prove it logically!” sneered Uncle Jim. “Fire away, then! I defy you to do it!”

“For the sake of argument,” Uncle Joe began, “let us assume Carr to be out. And let us see what that assumption would lead to. I’m going to do this by Reductio ad Absurdum.”

“Of course you are!” growled Uncle Jim. “Never knew any argument of yours that didn’t end in some absurdity or other!”

“Unprovoked by your unmanly taunts,” said Uncle Joe in a lofty tone, “I proceed. Carr being out, you will grant that, if Allen is also out, Brown must be at home?”

“What’s the good of his being at home?” said Uncle Jim. “I don’t want Brown to shave me! He’s too clumsy.”

“Patience is one of those inestimable qualities—” Uncle Joe was beginning; but Uncle Jim cut him off short.

“Argue!” he said. “Don’t moralise!” “Well, but do you grant it?” Uncle Joe persisted. “Do you grant me that, if Carr is out, it follows that if Allen is out Brown must be in?”

“Of course he must,” said Uncle Jim; “or there’d be nobody in the shop.”
Lewis Carroll Logic Paradox - continued

“We see, then, that the absence of Carr brings into play a certain Hypothetical, whose protasis is “Allen is out,” and whose apodosis is “Brown is in”. And we see that, so long as Carr remains out, this Hypothetical remains in force?”

“Well, suppose it does. What then?” said Uncle Jim.

“You will also grant me that the truth of a Hypothetical—I mean its validity as a logical sequence—does not in the least depend on its protasis being actually true, nor even on its being possible. The Hypothetical, “If you were to run from here to London in five minutes you would surprise people,” remains true as a sequence, whether you can do it or not.”

“I ca’n’t do it,” said Uncle Jim.

“We have now to consider another Hypothetical. What was that you told me yesterday about Allen?”

“I told you,” said Uncle Jim, “that ever since he had that fever he’s been so nervous about going out alone, he always takes Brown with him.” (¶ 26)

“Just so,” said Uncle Joe. “Then the Hypothetical, “if Allen is out Brown is out” is always in force, isn’t it?”

“I suppose so,” said Uncle Jim. (He seemed to be getting a little nervous, himself, now.)

“Then if Carr is out, we have two Hypotheticals, “if Allen is out Brown is in” and “If Allen is out Brown is out,” in force at once. And two incompatible Hypotheticals, mark you! They ca’n’t possibly be true together!”

“Ca’n’t they?” said Uncle Jim.

“How can they?” said Uncle Joe. “How can one and the same protasis prove two contradictory apodoses? You grant that the two apodoses, “Brown is in” and “Brown is out,” are contradictory, I suppose?”

“Yes, I grant that,” said Uncle Jim.

“Then I may sum up,” said Uncle Joe. “If Carr is out, these two Hypotheticals are true together. And we know that they cannot be true together. Which is absurd. Therefore Carr cannot be out. There’s a nice Reductio ad Absurdum for you!”

Uncle Jim looked thoroughly puzzled: but after a bit he plucked up courage, and began again. “I don’t feel at all clear about that incompatibility. Why shouldn’t those two Hypotheticals be true together? It seems clear to me that would simply prove “Allen is in”. Of course it’s clear that the apodoses of those two Hypotheticals are incompatible—“Brown is in” and “Brown is out”. But why shouldn’t we put it like this? If Allen is out Brown is out. If Carr and Allen are both out, Brown is in. Which is absurd. Therefore Carr and Allen ca’n’t be both of them out. But, so long as Allen is in, I don’t see what’s to hinder Carr from going out.”

“My dear, but most illogical, brother!” said Uncle Joe. (Whenever Uncle Joe begins to “dear” you, you may make pretty sure he’s got you in a cleft stick!) “Don’t you see that you are wrongly dividing the protasis and the apodosis of the Hypothetical? Its protasis is simply “Carr is out”; and its apodosis is a sort of sub-Hypothetical, “If Allen is out, Brown is in”. And a most absurd apodosis it is, being hopelessly incompatible with that other Hypothetical that we know is always true, “If Allen is out, Brown is out”. And it’s simply the assumption “Carr is out” that has caused this absurdity. So there’s only one possible conclusion. Carr is in!”

How long this argument might have lasted, I haven’t the least idea. I believe either of them could argue for six hours at a stretch. But, just at this moment, we arrived at the barber’s shop; and, on going inside, we found——
Lewis Carroll Logic Paradox - continued

Note.
The paradox, of which the forgoing paper is an ornamental presentation, is, I have reason to believe, a very real difficulty in the Theory of Hypotheticals. The disputed point has been for some time under discussion by several practised logicians, to whom I have submitted it; and the various and conflicting opinions, which my correspondence with them has elicited, convince me that the subject needs further consideration, in order that logical teachers and writers may come to some agreement as to what Hypotheticals are, and how they ought to be treated.
The original dispute, which arose, more than a year ago, between two students of Logic, may be symbolically represented as follows:—
There are two Propositions, $A$ and $B$.
It is given that
1. If $C$ is true, then, if $A$ is true, $B$ is not true;
2. If $A$ is true, $B$ is true.
The question is, can $C$ be true?
The reader will see that if, in these two Propositions, we replace the letters $A$, $B$, $C$ by the names Allen, Brown, Carr, and the words “true” and “not true” by the words “out” and “in” we get
1. If Carr is out, then, if Allen is out, Brown is in;
2. If Allen is out, Brown is out.
These are the very two Propositions on which “Uncle Joe” builds his argument.
Several very interesting questions suggest themselves in connexion with this point, such as
Can a Hypothetical, whose protasis is false, be regarded as legitimate?
Are two Hypotheticals, of the forms “If $A$ then $B$” and “If $A$ then not-$B$,” compatible?
What difference in meaning, if any, exists between the following Propositions?
1. $A$, $B$, $C$, cannot be all true at once;
2. If $C$ and $A$ are true, $B$ is not true;
3. If $C$ is true, then, if $A$ is true, $B$ is not true;
4. If $A$ is true, then, if $C$ is true, $B$ is not true.
The following concrete form of the paradox has just been sent me, and may perhaps, as embodying necessary truth, throw fresh light on the question.
Let there be three lines, $KL$, $LM$, $MN$, forming, at $L$ and $M$, equal acute angles on the same side of $LM$.
Let “$A$” mean “The points $K$ and $N$ coincide, so that the three lines form a triangle”.
Let “$B$” mean “The triangle has equal base-angles”.
Let “$C$” mean “The lines $KL$ and $MN$ are unequal”.
Then we have
1. If $C$ is true, then, if $A$ is true, $B$ is not true.
2. If $A$ is true, $B$ is true.
The second of these Propositions needs no proof; and the first is proved in Euc., i, 6, though of course it may be questioned whether it fairly represents Euclid’s meaning.
I greatly hope that some of the readers of Mind who take an interest in logic will assist in clearing up these curious difficulties.
The Barbershop Paradox

Categories:
Philosophy
June 3, 2011 by Tim

“The Barbershop Paradox” is one that is still debated today, and I have NO idea why. Today in this post we will discuss the problem, the debate, and why it is completely junk. This is not a paradox, it is another example of the “what happened to the other dollar?” question we have discussed.

From Wikipedia, the story is summarized as such:

Briefly, the story runs as follows: Uncle Joe and Uncle Jim are walking to the barber shop. There are three barbers who live and work in the shop—Allen, Brown, and Carr—but not all of them are always in the shop. Carr is a good barber, and Uncle Jim is keen to be shaved by him. He knows that the shop is open, so at least one of them must be in. He also knows that Allen is a very nervous man, so that he never leaves the shop without Brown going with him.

Uncle Joe insists that Carr is certain to be in, and then claims that he can prove it logically. Uncle Jim demands the proof. Uncle Joe reasons as follows.

Suppose that Carr is out. If Carr is out, then if Allen is also out Brown would have to be in—since someone must be in the shop for it to be open. However, we know that whenever Allen goes out he takes Brown with him, and thus we know as a general rule that if Allen is out, Brown is out. So if Carr is out then the statements “if Allen is out then Brown is in” and “if Allen is out then Brown is out” would both be true at the same time.

Uncle Joe notes that this seems paradoxical; the hypotheticals seem “incompatible” with each other. So, by contradiction, Carr must logically be in!

This a summary from a three page essay from Lewis Carroll called “A Logical Paradox” that was published in 1894 in Mind magazine. Due to it being part of public domain, clicking the link on the name will take you to the full essay if you want to read it.

The reasoning that this logic is sound is the belief that the statement “if Allen is out then Brown is in” and “if Allen is out then Brown is out” would both be true at the same time actually holds water. It doesn’t. If you go to wikipedia or any other number of discussions, you can find a whole slew of mathematical formulas that have no relevance whatsoever. Like most problems, if you don’t understand them, you make them much more difficult than they really are.

The truth is, this is much more related to the Monte Hall delima rather than Russell’s paradox. We all know that a person can not be in two places at the same time, so suggesting it is ridiculous, especially for someone who wants to speak of logic. Side note how can Sir Isaac Newton could figure out the three laws of motion, and the concept of calculated risk be so mangled 200 years later? Let me explain.
Lewis Carroll Logic Paradox - criticism

All that really happened here was outlining a set of possibilities. In the narrative it starts out with the following 8 possibilities:

- Allen is in, Brown is in, Carr is in
- Allen is out, Brown is in, Carr is in
- Allen is in, Brown is out, Carr is in
- Allen is in, Brown is in, Carr is out
- Allen is out, Brown is out, Carr is in
- Allen is out, Brown is in, Carr is out
- Allen is in, Brown is out, Carr is out
- Allen is out, Brown is out, Carr is out

Now the narrative starts setting conditions. First of all at least one of them has to be in. This is an acceptable condition. Someone must be in a place of business in order for it to operate. So now our 8 possibilities are down to 7 and look like this:

- Allen is in, Brown is in, Carr is in
- Allen is out, Brown is in, Carr is in
- Allen is in, Brown is out, Carr is in
- Allen is in, Brown is in, Carr is out
- Allen is out, Brown is out, Carr is in
- Allen is out, Brown is in, Carr is out
- Allen is in, Brown is out, Carr is out
- Allen is out, Brown is out, Carr is out

I (DrCannata) added the red lines because “Carr is out” is being assumed. Also, I moved the second black line from the bottom down 1 because it was in error.
Lewis Carroll Logic Paradox - criticism

The next set of conditions set is that if Allen is out, he takes Brown with him. In the narrative this is due to a medical condition. Whether or not it makes sense is immaterial really, we really just have to accept this one. So what this eliminates from the list is any instance in which Allen is out, and Brown is in. Moving out 7 possibilities, down to five. Now they look like this.

Allen is in, Brown is in, Carr is in
Allen is out, Brown is in, Carr is in
Allen is in, Brown is out, Carr is in
Allen is in, Brown is in, Carr is out
Allen is out, Brown is out, Carr is in
Allen is out, Brown is in, Carr is out
Allen is in, Brown is out, Carr is out
Allen is out, Brown is out, Carr is out

Now with the possible outcomes above, there is a 3/5 chance that Carr is in. Because only one of these possibilities can exist at any given time. The item suggested that “if Allen is out then Brown is in” and “if Allen is out then Brown is out” would both be true at the same time simply cannot be because Brown cannot be in the barbershop, and somewhere else at the same time. Now we can accept the possibility of “if Allen is out then Brown is in” not existing, but the idea that because one is true, the other is as well has no form of logic in it at all.

Now if this article is suppose to hint at some greater truth that was trying to be explained through story I have no idea. I do know that if it was, they should have made up a better story to support it. Although, if the underlying item being presented is that if one possibility can’t exist, neither can another, I know that it is that kind of close-mindedness that creates the paradox of the human condition. It saddens me that seeds like this were planted all so many years ago. In essence, that thinking is an attempt to giving logical justification for the bigottedness that plagues societies the world over. It’s why we must correct these thought processes.
Bells (Inequality) Theorem

/* Bell’s Theorem - num(disagree1) + num(disagree2) >= num(disagree3)
 * 
 * If see
 * 0 30 60
 * a_strategy: { P, P, P }
 * b_strategy: { A, P, P }
 * c_strategy: { P, A, P }
 * d_strategy: { P, P, A }
 */

disagree1(N) :- b_strategy(N), data(N,0,30).
disagree1(N) :- b_strategy(N), data(N,30,0).
disagree1(N) :- c_strategy(N), data(N,0,30).
disagree1(N) :- c_strategy(N), data(N,30,0).
disagree2(N) :- d_strategy(N), data(N,30,60).
disagree2(N) :- d_strategy(N), data(N,60,30).
disagree2(N) :- c_strategy(N), data(N,30,60).
disagree2(N) :- c_strategy(N), data(N,60,30).
disagree3(N) :- d_strategy(N), data(N,0,60).
disagree3(N) :- d_strategy(N), data(N,60,0).
disagree3(N) :- d_strategy(N), data(N,0,60).
disagree3(N) :- d_strategy(N), data(N,60,0).

data(1,0,0).
data(2,0,30).
data(3,0,60).
data(4,30,0).
data(5,30,30).

/* This doesn't seem to work unfortunately */
discontiguous(b_strategy).
discontiguous(c_strategy).
discontiguous(d_strategy).

* so the following are in sorted order instead of number order of the arguments.
*/

b_strategy(1).
b_strategy(4).
c_strategy(5).
c_strategy(2).
d_strategy(3).
Alain Aspect (French pronunciation: [aspɛ] (help·info); born 15 June 1947 in Agen) is a French physicist and alumnus of the École Normale Supérieure de Cachan (ENS Cachan) in France. In the early 1980s, while working on his PhD thesis from the lesser academic rank of lecturer, he performed the elusive "Bell test experiments" that showed that Albert Einstein, Boris Podolsky and Nathan Rosen's reductio ad absurdum of quantum mechanics, namely that it implied 'ghostly action at a distance', did in fact appear to be realised when two particles were separated by an arbitrarily large distance. A correlation between their wave functions remained, as they were once part of the same wave-function that was not disturbed before one of the child particles was measured.

If quantum theory is correct, the determination of an axis direction for the polarization measurement of one photon, forcing the wave function to 'collapse' onto that axis, will influence the measurement of its twin. This influence occurs despite any experimenters not knowing which axes have been chosen by their distant colleagues, and at distances that disallow any communication between the two photons, even at the speed of light.

Aspect's experiments were considered to provide overwhelming support to the thesis that Bell's inequalities are violated in its CHSH version. However, his results were not completely conclusive, since there were so-called loopholes that allowed for alternative explanations that comply with local realism. See local hidden variable theory.

Stated more simply, the experiment provides strong evidence that a quantum event at one location can affect an event at another location without any obvious mechanism for communication between the two locations. This has been called "spooky action at a distance" by Einstein (who doubted the physical reality of this effect). However, these experiments do not allow faster-than-light communication, as the events themselves appear to be inherently random.
Bells (Inequality) Theorem

John Stewart Bell (1928-1990) was a brilliant Irish physicist. He was only seven years old when the EPR paradox was published. In his early 20s, Bell became fascinated by this scientific controversy. In the beginning, he was a strong supporter of Einstein's view, stating that:

*I felt that Einstein's intellectual superiority over Bohr, in this instance, was enormous; a vast gulf between the man who saw clearly what was needed, and the obscurantist.*

A bizarre twist lay in the future.

By the 1960's, Bell's day job was working as a particle physicist and accelerator designer at CERN. However, his part-time obsession was quantum theory, especially with the "hidden variables" mentioned in the EPR paradox. In 1963, Bell took a year's leave from CERN and was able to spend significantly more time on his pet obsession. He focused on working out a way to experimentally test the existence of these hidden variables. Finally, he published Bell's Inequality (or Bell's Theorem) in 1964. Bell's inequality makes it possible to construct experiments to directly test if "spooky action at a distance" actually occurs.

Other physicists began to perform experiments based on his hypothesis. After numerous experiments using many different approaches, the result is clear. Quantum entanglement is real. There is no need to postulate any hidden variables to account for the results.

John Bell had single-handedly disproved the great Albert Einstein.

The parallels could not be more obvious - just as Einstein had once rendered the "luminiferous aether" irrelevant using the theory of relativity, so Bell has also rendered Einstein's "hidden variables" irrelevant using Bell's inequality. Bell's inequality was hailed by fellow physicists as the *most profound discovery in science*. In the late 1980s, Bell was even nominated for a Nobel Prize.

Unfortunately, Bell would never enjoy anywhere near the amount of fame and prestige that Einstein received. On the 1st of October 1990, John Bell died suddenly of a stroke. Had he lived longer, he would most likely have been awarded the Physics prize, and things might have turned out very differently.

Alas, because theoretical physics was inaccessible to the public at that time, most people never came to understand the importance of his work. And so John Stewart Bell, the part-timer who proved Einstein wrong, was consigned to the footnotes of history.
**Haskell**

head :: [a] -> a
head (x : _) = x

tail :: [a] -> [a]
tail (_ : xs) = xs

null :: [a] -> Bool
null [] = True
null (_ : _) = False

lastelem :: [a] -> a
lastelem [x] = x
lastelem (_ : xs) = lastelem xs

initelem :: [a] -> [a]
ineitelem [ _ ] = []
initelem (x : xs) = x : initelem xs

listlength :: [a] -> Int
listlength [] = 0
listlength (_ : l) = 1 + listlength l

sumList :: (Num a) => [a] -> a
sumList [] = 0
sumList (x : xs) = x + sumList xs

append :: [a] -> [a] -> [a]
append [] ys = ys
append (x : xs) ys = x : append xs ys

**Prolog**

head( [ X | _ ], X).

tail( [ _ | Xs ], Xs).

null( [] ).

lastelem( [ X ], X).
lastelem( [ _ | Xs ], Y) :- lastelem(Xs, Y).

initelem( [ _ ], [ ]).
initelem( [ X | Xs ], [ X | Ys ]):- initelem(Xs, Ys).

listlength( [ ], 0).
listlength( [ _ | L ], N) :- listlength(L, N0), N is 1+N0.

sumList( [ ], 0).
sumList( [ X | Xs ], N):- sumList(Xs, N0), N is X+N0.

appendList( [ ], Ys, Ys).
appendList( [X | Xs], Ys, [X | Zs]) :- appendList(Xs, Ys, Zs).

---

<table>
<thead>
<tr>
<th>Clause</th>
<th>Conventional Syntax</th>
<th>PROLOG Terminology</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ p }</td>
<td>p</td>
<td>facts</td>
</tr>
<tr>
<td>{ q }</td>
<td>q</td>
<td></td>
</tr>
<tr>
<td>{ r }</td>
<td>r</td>
<td></td>
</tr>
<tr>
<td>{ s, ~p, ~q }</td>
<td>p \land q \rightarrow s</td>
<td>s:p,q.</td>
</tr>
<tr>
<td>{ t, ~s, ~r }</td>
<td>s \land r \rightarrow t</td>
<td>t:s,r.</td>
</tr>
<tr>
<td>{ ~t }</td>
<td>~t \rightarrow \bot</td>
<td>\neg t.</td>
</tr>
</tbody>
</table>

This Prolog Code can be found in 11Prolog Examples.p