1 First Order Logic Cont.

1.1 Modus Ponens

If $A \Rightarrow B$, read A implies B, and we know A is true, then we know B is true. This is because the truthfulness of A *implies* that B is also true given the nature of the problem. For example, the sprinklers being on implies the ground is wet. Note, the group being wet does not imply the sprinklers are on. it could be raining.

1.2 Contraposition

The rule of contraposition states that is $A \Rightarrow B$, then $Not(B) \Rightarrow Not(A)$. Stated more plainly, we know that if B is not truthful, then A also must not be truthful, because if A were truthful, then it would imply B.

For example: All dogs are mammals. If something is a dog it implis it is also a mammal. The contrapositive is, "If something is not a mammal, it cannot be a dog".

1.3 Example

Please walk through slide 16 from the Logic slides to see how to formalize a proof using these rules from above.

1.4 Conjunctive Normal Form (CNF)

Any knowledge base can be written as a single formula in conjunctive normal form (CNF). The proof is in the book for those interested.

1.4.1 Form Structure

CNF formula: (... OR ... OR ...) AND (... OR ...) AND ...
Note: can include variables and the NOT of variables

1.4.2 Example

(A OR NOT(B) OR C OR NOT(D)) AND (C OR D OR NOT(A))

Any assignment where C is true will satisfy this form, since C is in both clauses. As will anything with (A and D), and so on.

1.4.3 Unit Resolution

Unit resolution is very similar to modus ponens, but is applicable to CNF form statements. It states that if you know one of the OR variabels is false, you can drop it from the clause.

For example. If we have (A OR B OR C) and we know NOT(B), then we can rewrite the original clause as (A OR C), since we know B is false. Additionally if you had started with (A OR NOT(B) OR C) and you knew B was true then you can again simplify to (A OR C).

An example of a resolution proof can be found n slide 21 of the Propositional Logic slides.

1.4.4 Limitations of Unit Resolution

Consider the following case:

A OR B
NOT(A) OR B

We know B must be true. Because if A if false, then B is needed to satisfy A OR B and if A is true, then B is needed to satisfy NOT(A) OR B.

1.5 (General) Resolution

If we have:

$L_1 \quad OR \quad L_2 \quad OR \quad L_3 \quad ... \quad OR \quad L_n$
and
$K_1 \quad OR \quad K_2 \quad OR \quad K_3 \quad ... \quad OR \quad K_m$

And for some (i,j) where $1 \leq i \leq n$ and $1 \leq j \leq m$ such that $L_i = NOT(K_j)$, then we can conclude:

$L_1 \quad ... \quad OR \quad L_{i-1} \quad OR \quad L_{i+1} \quad ... \quad OR \quad L_n \quad OR \quad K_1$
$... \quad OR \quad K_{j-1} \quad OR \quad K_{j+1} \quad ... \quad OR \quad K_m$

2 Search Cont.

2.1 Linear Programming

Linear programming is a technique used to find the best outcome in a mathematical model whose requirements are represented by linear relationships.

2.2 Parts

A linear program definition is comprised of two parts. An objective, and constraints. The output of a linear program solver will be the maximum result achievable for the objective without breaking any of the constraints. In a setting where you wanted to minimize, not maximize the objective, you would multiply it by a factor -1, and then maximize.