A Natural Deduction System

This system is based on one by Willard Van Orman Quine. A proof in this system is a sequence of lines. Each line is numbered, has zero or more stars, a formula and a justification. The stars are used to keep track of the scope of assumptions that have been made during the proof. If the last line has no stars, the proof is said to be a proof of the formula in that line.

Propositional rules

Premise rule: Any formula can be added as a line in the proof by this rule. If the line is the first line in the proof, its number will be 1, there will be one star. If it is line j+1, its number will be j+1, it will have one more star than line j has, and in both cases, the justification is **pr**. (A premise is the same as an assumption.)

Conditionalization rule: Let line j have k stars. Let line i be the first line with k stars such that every line between i and j has k or more stars. (In other words, line i through line j is the scope of the assumption that was introduced in line i.) If `A` is the formula in line i and `B` is the formula in line j, add line j+1 with formula `(A --> B)`, k-1 stars, and justification **cd**. (This is essentially the same as the Deduction rule in the book.)

Truth-functional rule: Suppose that A_{i1}, ..., A_{ik} are the formulas on eligible lines numbered i1, ..., ik, all of which are less than or equal to j, and { A_{i1}, ..., A_{ik}} |= `B`; add line j+1 with formula `B`, the same number of stars as line j, and justification **tf** i1, i2, ..., ik. Line number i is *eligible* if it has no more stars than line j has and every line between i and j has at least as many stars as line i has. (In other words, line j is in every scope that line i is in.)

The first line of the proof can be introduced by the truth-functional rule. In this case, the line number is 1, there are no stars, the formula is a tautology, and the justification is just **tf**. (A tautology can be introduced anywhere else in the proof too, since it is a logical consequence of the empty set.)

The truth-functional rule is deliberately powerful so that much can be inferred in one step when we get to doing proofs in predicate logic. When we are only proving tautologies, however, we will limit the use of the truth-functional rule to instances based only on the following logical consequences.

```
{A,B} |= (A ^ B)              {(A ^ B)} |= A               {(A ^ B)} |= B

{A} |= (A v B)                {B} |= (A v B)               {(A v B),(A --> C),(B --> C)} |= C

{A,(A --> B)} |= B            {(A --> (B ^ ~B))} |= ~A     {(A ^ ~A)} |= B

{~~A} |= A                    {A} |= ~~A                   |= (A v ~A)

{(A <-> B)} |= (A --> B)      {(A <-> B)} |= (B --> A)     {(A --> B),(B --> A)} |= (A <-> B)
```

(These correspond to individual rules in most other natural deduction systems.)

An example of the proof of a tautology using this restricted version of the truth-functional rule:

We prove the tautology `((p ^ q) --> ~(~q v ~p))`.

```
1.  *    (p ^ q)                              pr
2.  **    (~q v ~p)                           pr
3.  ***    ~q                                 pr
4.  ***    q                                  tf 1
5.  ***    (q ^ ~q)                           tf 3,4
6.  **    (~q --> (q ^ ~q))                   cd
7.  ***    ~p                                 pr
8.  ***    p                                  tf 1
9.  ***    (p ^ ~p)                           tf 7,8
10. ***    (q ^ ~q)                           tf 9
11. **    (~p --> (q ^ ~q))                   cd
12. **    (q ^ ~q)                            tf 2,6,11
13. *    ((~q v ~p) --> (q ^ ~q))             cd
14. *    ~(~q v ~p)                           tf 13
15.     ((p ^ q) --> ~(~q v ~p))              cd
```

Quantifier rules

We will need to introduce free variables from time to time in proofs that involve quantified formulas. For each natural number *n*, let `x`*n* be the nth variable symbol that we will use as a free variable. These variables will be distinct from the variables that are quantified (the latter will not have numbers in the symbols). To simplify typing the quantifiers, `A` will be used for the universal quantifier symbol (upside-down A), and `E` will be used for the existential quantifier symbol (backwards E).

Universal Instantiation rule: Suppose that line j has k stars and the formula for some eligible line i ≤ j has the form `AxF(x)`. Add line j+1 with k stars and formula `F(z)` where `z` is either a constant or one of the free variable symbols mentioned above. The justification for line j+1 is UI i.

Existential Generalization rule: Suppose that line j has k stars and the formula for some eligible line i ≤ j has the form `F(z)` where `z` is a constant or one of the free variable symbols. Add line j+1 with k vertical bars, formula `ExF(x)` and justification EG i. The variable `x` should not have a number in its symbol, every free occurrence of `x` in `F(x)` should correspond to an occurrence of `z` in `F(z)` and every occurrence of `z` in `F(z)` that got replaced was replaced by a free occurrence of `x`. (Note that not every occurrence of `z` in `F(z)` need be replaced by `x`, and it is rare that an occurrence of `z` is not replaced.)

Existential Instantiation rule: Suppose that line j has k stars and the formula for some eligible line i ≤ j has the form `ExF(x)`. Add line j+1 with k stars, the formula `F(x`*n*`)` and justification EI i `x`*n*. The index *n* must be larger than the index of any other free variable in lines 1 through j+1.

Universal Generalization rule: Suppose that line j has k stars and the formula for some eligible line i ≤ j has the form `F(x`*n*`)`. Add line j+1 with k stars, formula `AxF(x)` and justification UG i `x`*n*. The index *n* must be larger than the index of any other free variable in `F(x`*n*`)`. The symbol `x` should not contain a number, every occurrence of `xn` in `F(xn)` should correspond to a free occurrence of `x` in `F(x)`, and every free occurrence of `x` in `F(x)` should correspond to an occurrence of `xn` in `F(xn)`.

Important: The free variable `xn` is said to be **flagged** when it appears in a line justification. A free variable may be flagged only **once** in the entire proof. Furthermore, when using the **cd** rule to infer a formula P --> Q, none of the free variables in sub-formula P should be flagged anywhere in the proof between line i, which has formula P, and line j, which has formula Q . This prevents deduction of formulas like F(x1) --> AxF(x), which is not a logically valid formula.

An example of a proof using the quantifier rules.

We prove the formula `(ExAyF(x,y) --> AyExF(x,y))`.

```
1. *  ExAyF(x,y)                         pr
2. *  AyF(x1,y)                          EI 1 x1
3. *  F(x1,x2)                           UI 2
4. *  ExF(x,x2)                          EG 3
5. *  AyExF(x,y)                         UG 4 x2
6. (ExAyF(x,y) --> AyExF(x,y))           cd
```

Note that the implication in the other direction cannot be proved. The step equivalent to going backwards from line 3 to line 2 above would violate the UG rule.

Another example: We prove `((AxEy(F(x) --> G(x,y)) ^ (ExEyG(x,y) --> Ax~F(x))) --> Ax~F(x))`.

```
1. *  (AxEy(F(x) --> G(x,y)) ^ (ExEyG(x,y) --> Ax~F(x)))              pr
2. **  F(x1)                                                          pr
3. **  AxEy(F(x) --> G(x,y))                                          tf 1
4. **  Ey(F(x1) --> G(x1,y))                                          UI 3
5. **  (F(x1) --> G(x1,x2))                                           EI 4 x2
6. **  G(x1,x2)                                                       tf 2, 5
7. **  EyG(x1,y)                                                      EG 6
8. **  ExEyG(x,y)                                                     EG 7
9. **  Ax~F(x)                                                        tf 1, 8
10.**  ~F(x1)                                                         UI 9
11.*  (F(x1) --> ~F(x1))                                              cd
12.*  ~F(x1)                                                          tf 11
13.*  Ax~F(x)                                                         UG 12 x1
14.((AxEy(F(x) --> G(x,y)) ^ (ExEyG(x,y) --> Ax~F(x))) --> Ax~F(x))   cd
```

A third example: We prove `Ax(B(x) ^ C(x)) --> (AxB(x) ^ AxC(x))`

```
1.  * Ax(B(x) ^ C(x))                                                 pr
2.  * (B(x1) ^ C(x1))                                                 UI 1
3.  * B(x1)                                                           tf 2
4.  * AxB(x)                                                          UG 3 x1
5.  * (B(x2) ^ C(x2))                                                 UI 1
6.  * C(x2)                                                           tf 5
7.  * AxC(x)                                                          UG 6 x2
8.  * (AxB(x) ^ AxC(x))                                               tf 4, 7
9.  Ax(B(x) ^ C(x)) --> (AxB(x) ^ AxC(x))                             cd
```