Logic: logic.py; Chapters 6-8

We'll start by looking at Expr, the data type for logical sentences, and the convenience function expr. We'll be covering two types of knowledge bases, PropKB - Propositional logic knowledge base and FolKB - First order logic knowledge base.

But the first step is to load the code:

In [1]:

from utils import \*

from logic import \*

Logical Sentences

The Expr class is designed to represent any kind of mathematical expression. The simplest type of Expr is a symbol, which can be defined with the function Symbol:

In [2]:

Symbol('x')

Out[2]:

x

Or we can define multiple symbols at the same time with the function symbols:

In [3]:

(x, y, P, Q, f) = symbols('x, y, P, Q, f')

We can combine Exprs with the regular Python infix and prefix operators. Here's how we would form the logical sentence "P and not Q":

In [4]:

P & ~Q

Out[4]:

(P & ~Q)

This works because the Expr class overloads the & operator with this definition:

In [5]:

sentence = P & ~Q

sentence.op

Out[5]:

'&'

In [6]:

sentence.args

Out[6]:

(P, ~Q)

In [7]:

P.op

Out[7]:

'P'

In [8]:

P.args

Out[8]:

()

In [9]:

Pxy = P(x, y)

Pxy.op

Out[9]:

'P'

In [10]:

Pxy.args

Out[10]:

(x, y)

## Operators for Constructing Logical Sentences

A screenshot of a computer

Description automatically generated with medium confidence

In [13]

expr('~(P & Q) ==> (~P | ~Q)')

## expr: a Shortcut for Constructing Sentences

If the |'==>'| notation looks ugly to you, you can use the function expr instead:

In [14]

expr('~(P & Q) ==> (~P | ~Q)')