CS402 cw1 report

20150186 김재욱

I'm really really not good at English so if you don't understand my sentence please message me

1. CNF Converter

usage: python cnf.py "polish notation"

I made full binary tree of formula. Each node has symbol, left, right and neg attributes.
- symbol : one operators or literals
- left, right : child of this node. All operators without negation are binary operator so if symbol of this node is operator then it has left and right, in. else case (symbol is literal) it doesn't have childs
- neg : present whether this node is negative or not

• parsing
read symbols from back.
if symbol is operator, pop two element and set them current symbol's child

• convert
convert imply and equivalent to combine of *and*, *or*, and *negation* recursively
then
make operator node don't have negation recursively.
then
make cnf using distribute rule recursively

• print prefix and infix
prefix : parent left right
infix : left parent righte
using this order and recursively print

• validity check
cnf = clause & clause & clause & … & clause
if p and -p in one clause, the clause is True
so make dict that dict[p]=(has_negations?)
and check if there are key in dict but original boolean value isn't same new value, the cluase is True.
then check all clause are True, it is valid.

2. CNF Converter

usage: set CWD file in same directory of nonogram.py then
python nonogram.py

• how to encoding
see one line, size: 6 and rule: 2 1
there are 6 possible cases:
ooxoxx xooxox
ooxxox xooxxo
ooxxxo xxooxo

naming proposition value v1~v6.
the one line logical formula become v1 or v2 or … or v6
naming other line logical formula same way.
For all interchange of n*m lines add rule about can't exist simultaneously.
example)
row1 is size:6 and rule: 2 1
col1 is size:6 and rule: 1
there are 6 cases (see upper side) of row1
r1 = ooxoxx
r2 = ooxxox
r3 = ooxxxo
r4 = xooxox
r5 = xooxxo
r6 = xxooxo
there are 6 cases of col1 : c1 ~ c6
c1 = oxxxxx
c2 = xoxxxx
c3 = xxoxxx
c4 = xxxoxx
c5 = xxxxox
c6 = xxxxxo
then (1,1) True rule is (r1,r2,r3,c1) and False rule is (r4,r5,r6,c2,c3,c4,c5,c6)
True rule and False rule can't exist simultaneously
so add rule [-t and -f] (it is same t -> -f, f -> -t) of t in True rule and f in False rule