4.1

Allgemeiner Code, der für alle folgenden Aufgaben verwendet wurde:

Python console for SymPy 1.0 (Python 2.7.12)

These commands were executed:

>>> from \_\_future\_\_ import division

>>> from sympy import \*

>>> x, y, z, t = symbols('x y z t')

>>> k, m, n = symbols('k m n', integer=True)

>>> f, g, h = symbols('f g h', cls=Function)

>>> a, b, c, d = symbols("a, b, c, d")

>>> def implication( x, y):

... return satisfiable( Not(Implies( x, y ))) == False

>>> def equivalence( z, t ):

... return implication( z, t) & implication( t, z )

a) i)

>>> phi = (Not(a) | b) & (Not(b) | c) & (Not(c) | d) & (Not(d) | a)

>>> psi = 1 ^ a ^ b ^ c ^ d

>>> implication( phi, psi )

True

>>> implication( psi, phi )

False

>>> equivalence( phi, psi )

False

ii)

>>> phi = Not(a | b | c) | (Not(a) & b & c) | (a & (b ^ c))

>>> psi = (a | (b ^ c)) & (a >> Equivalent(b, c))

>>> implication( phi, psi )

False

>>> implication( psi, phi )

False

>>> equivalence( phi, psi )

False

b) i)

>>> alpha = ((a >> b) & (b >> c))

>>> beta = (a >> c)

>>> implication( alpha, beta )

True

ii)

>>> gamma = ((a >> b) & b)

>>> implication( gamma, a )

False

iii)

Verwendung von neuen Funktionen (allg. Code gilt nicht mehr)

>>> a, b, c, p, q = symbols("a, b, c, p, q")

>>> def implication(x, y):

... return satisfiable( Not(Implies( x, y ))) == False

>>> def equivalence( p, q, z):

... return implication(p, q) & implication(q, p) & implication(q, z) & implication(z, q) & implication(p, z) & implication(z, p)

>>> Equivalent((satisfiable(Not(Equivalent(a, b, c))) == False), (satisfiable(Not(equivalence(a, b, c))) == False))

True

>>> satisfiable(Not(Equivalent(a, b, c))) == False

False

>>> satisfiable(Equivalent(a, b, c)) == False

False

4.4

b)

Python console for SymPy 1.0 (Python 2.7.12)

These commands were executed:

>>> from \_\_future\_\_ import division

>>> from sympy import \*

>>> x, y, z, t = symbols('x y z t')

>>> k, m, n = symbols('k m n', integer=True)

>>> f, g, h = symbols('f g h', cls=Function)

>>> a, b, c, d, e, f, g, h = symbols("a, b, c, d, e, f, g, h")

>>> psi1 = Equivalent(h, (Or(Not(e), c)))

>>> psi2 = e >> Or(a, b, c, d, f, g, h)

>>> psi3 = b & Or(Not(g), Not(c))

>>> psi4 = Xor(f, c)

>>> psi5 = Not(Xor(a, b, c, d, e, f, g, h))

>>> psi6 = Xor(a, c)

>>> psi = And(phi1, phi2, phi3, phi4, phi5, phi6)

c)

>>> satisfiable(And(psi, Not(d)))

{a:True,b:True,c:False,d:False,e:False,f:True,g:False,h:True}

>>> for x in satisfiable(psi, all\_models=True):

... pretty\_print(x)

{a: True, b: True, c: False, d: False, e: False, f: True, g: False, h: True}

{a: True, b: True, c: False, d: True, e: False, f: True, g: True, h: True}

{a: True, b: True, c: False, d: False, e: True, f: True, g: False, h: False}

{a: True, b: True, c: False, d: True, e: True, f: True, g: True, h: False}

{a: False, b: True, c: True, d: True, e: False, f: False, g: False, h: True}

{a: False, b: True, c: True, d: False, e: True, f: False, g: False, h: True}

d)

i)

>>> satisfiable(And(phi, Or(e, f, h)))

{a:True,b:True,c:False,d:False,e:False,f:True,g:False,h:True}

ii)

>>> em, fm, fy, hm, ay, gy = symbols("em, fm, fy, hm, ay, gy")

>>> psi\_e = em >> e

>>> psi\_f = Or(fm, fy) >> f

>>> psi\_h = hm >> h

>>> psi\_a = ay >> a

>>> psi\_g = gy >> g

>>> psi\_f2 = And((fy >> Not(fm)), (fm >> Not(fy)))

>>> psi\_M = Or(em, fm, hm)

>>> psi\_Y = Or(ay, fy, gy)

>>> psi\_all = And(psi\_e, psi\_f, psi\_h, psi\_a, psi\_g, psi\_f2, psi\_M, psi\_Y)

>>> satisfable(psi\_all)

>>> satisfiable(And(phi, psi\_all))

{a:True,ay:True,b:True,c:False,d:False,e:False,em:False,f:True,fm:False,fy:False,g:False,gy:False,h:True,hm:True}