

My Robot Chemist Knows How to Draw Chemical Structure

I am an organic chemist, and my daily work is synthesizing novel small molecules as potential drug candidates. I am going to design an AI agent that can draw chemical structures based on formulas using the concept **Constraint Propagation** learned in KBAI class.

The challenges

From a chemist's view, the world is built on molecules. A molecule is the smallest particle in a compound that has chemical properties of this compound, and is an electrically neutral group of atoms connected through chemical bonds. The chemical formula provides information about proportions of various atoms in a molecule. A simple example is given in Figure 1. Ibuprofen (trade name as Advil, Motrin) is composed of 13 carbon atoms, 18 hydrogen atoms and 2 oxygen atoms. Its chemical structure is shown in Figure 1. Different molecules may have same proportions of atoms (same chemical formula), but with different arrangements (connections) between atoms (different chemical structure). They are called isomers, as shown in Figure 2a,b. However, in the chemical structure, atoms

are not randomly connected. People without chemistry background may draw a chemical structure that makes nonsense, even

though the formula matches (Figure 2c). Here, I present how to use *Constraint Propagation* to design AI agent that can read a formula and draw a valid chemical structure.

Figure 1

Advil (ibuprofen)

Formula $C_{13}H_{18}O_2$

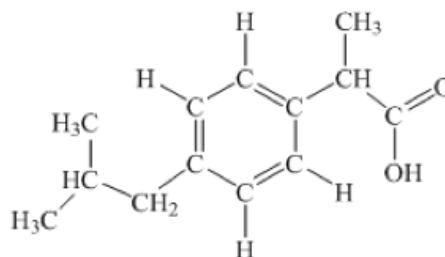
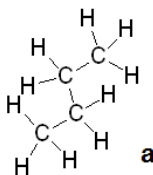

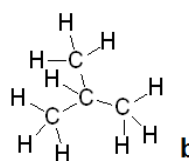



Figure 2

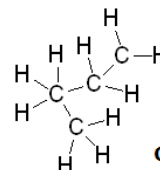
Formula C_4H_{10}




isomer 1 



isomer 2 

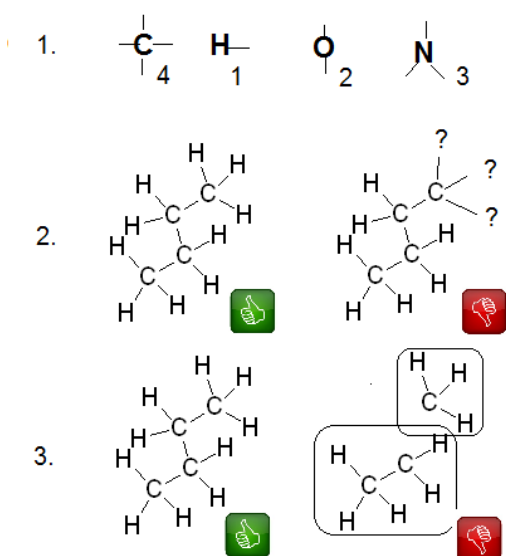


wrong structure 

Constraints in chemical structures

In KBAI class, constraint propagation is defined as: a method of inference that assigns values to variables characterizing a problem in such a way that some conditions (called constraints) are satisfied. So what are constraints in a chemical structure? Based on high school chemistry education, we know there are three constraints in a chemical structure (Figure 3).

Figure 3



Constraint 1: different atoms have different valences (bonds formed with others), as carbon atom can form 4 bonds with other atoms, while hydrogen atom can only form 1 bond with other atom.

Constraint 2: in a molecule, there should be no open bonds (each bond should connect two atoms).

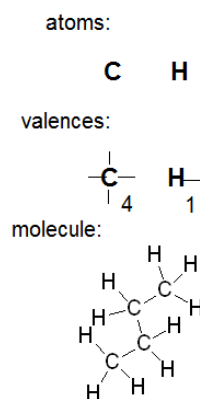
Constraint 3: in a molecule, all atoms should be connected together.

Implementation

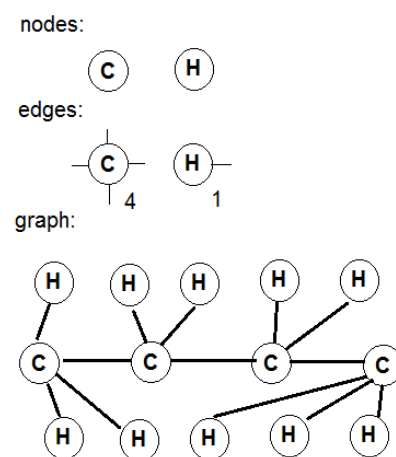
Chemists and computer scientists may have different views when they see a chemical structure (Figure 4). The chemist sees the molecule formed by atoms. The atoms (carbon or hydrogen) have their unique properties (property of valences is one of them), and the molecule also has its chemical properties (one kind of alkane, can

Figure 4

Chemist's view:



Computer scientist's view:



burn in air generating carbon dioxide and water). While, the computer scientist may say I don't need to know their chemical properties, and with the provided constraints I already can design an AI agent to draw correct chemical structures. The computer scientist may simplify atoms as different nodes, the valences as maximal edges of one specific node, the molecule as a graph meets some criteria (each node's edge number reaches its maximum; all nodes in the graph are connected).

Using the same strategy, I implemented an AI agent that can draw chemical structures of hydrocarbons based on their formulas (see attached python code). I created a class called *Atom*, which has attributes as valences, left valences, and adjacent atoms (stored in a list). I also created two help functions. Function *connect* connects two atom objects, appends these two objects to each other's adjacent lists and return a Boolean value. Function *hydrocarbon* draws a valid chemical structure (by print out the connections between atoms) based on a given chemical formula (as a python dictionary: {'C':4, 'H':10}). One example is provided in Figure 5.

```
-----  
There is a structure fits formula: C6H6  
C1 connects: [C2, C6, H1, H2]  
C2 connects: [C1, C3, C6, C3]  
C3 connects: [C2, C4, C2, H3]  
C4 connects: [C3, C5, C5, H4]  
C5 connects: [C4, C4, C6, H5]  
C6 connects: [C1, C2, C5, H6]
```

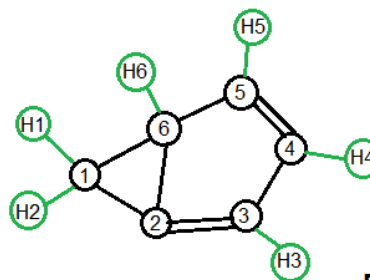


Figure 5

Limitations

The current version gives valid chemical structures of hydrocarbons (only contain carbon and hydrogen). But this agent can be upgraded to include other atoms as oxygen, nitrogen. My agent has a defect that although it gives valid chemical structure which satisfies constraints listed in Figure 3, it cannot optimize the result and deliver the best one. For example, to find a chemical structure fits formula C6H6, most chemists would think of benzene instead of the chemical structure shown in Figure 5. To them, the three membered carbon ring structure contains too much strain and is unstable. To make my robot chemist smarter, I have to add more constraints during implementation. A sophisticated score system may help.

```
import random
class Atom(object):
    def __init__(self, name, element):
        valence = {'C':4, 'H':1}
        self.name = name
        self.val = valence[element] #total valences
        self.lval = valence[element] #open valences
        self.adjacent = []
    def __repr__(self):
        return self.name

def connect(atom1, atom2):
    if atom1.lval == 0 or atom2.lval == 0:
        return False
    else:
        atom1.lval -= 1
        atom1.adjacent.append(atom2)
        atom2.lval -= 1
        atom2.adjacent.append(atom1)
        return True

def hydrocarbon(formula): # formula = {'C':4, 'H':10}
    carbonList = []
    hydrogenList = []
    molecule = []

    #transfer formula into lists of carbon/hydrogen atoms
    for i in range(formula['C']):
        carbonList.append(Atom('C'+str(i+1), 'C'))
    for j in range(formula['H']):
        hydrogenList.append(Atom('H'+str(j+1), 'H'))
    #molecule start with one carbon atom
    molecule.append(carbonList[0])
    #connect carbons in molecule in a random order
    for i in range(1, len(carbonList)):
        switch = True
        while switch:
            if connect(carbonList[i], random.choice(molecule)):
                molecule.append(carbonList[i])
                switch = False
    #calculate total open valences
    lv_sum = 0
    for carbon in molecule:
        lv_sum += carbon.lval
    while True:
        if lv_sum < len(hydrogenList):
            print '-----'
            print 'No structure fits formula: ' + 'C' + str(formula['C']) + 'H' + str(formula['H'])
            return None
        elif lv_sum > len(hydrogenList):
            r1 = random.randint(0, len(molecule)-1)
```

```
    r2 = random.randint(0, len(molecule)-1)
    if r1 != r2:
        if connect(molecule[r1], molecule[r2]):
            lv_sum -= 2
    else:
        #connect hydrogen in molecules
        for hydrogen in hydrogenList:
            for carbon in molecule:
                if connect(hydrogen, carbon):
                    break

        #print connections between carbon and hydrogen
        print '-----'
        print 'There is a structure fits formula: ' + 'C' + str(formula['C']) + 'H' + str(formula['H'])
        for carbon in molecule:
            print carbon.name, ' connects: ', carbon.adjacent
        return None
```

```
formula1 = {'C':4, 'H':10}
formula2 = {'C':6, 'H':6}
formula3 = {'C':2, 'H':4}
formula4 = {'C':2, 'H':2}
formula5 = {'C':2, 'H':3}
hydrocarbon(formula1)
hydrocarbon(formula2)
hydrocarbon(formula3)
hydrocarbon(formula4)
hydrocarbon(formula5)
```