Middle East Technical University | Department of Computer Engineering

CENG 242

Programming Language Concepts

Spring '2018-2019

Programming Assignment 5

Due date: May 22, Wednesday, 2019, 23:55
Version: 1.0

1 Introduction

In this assignment, you are going to use Prolog in order to define several rules of Chemistry. Our focus will be on basics of atoms and chemical bonds. For the sake of simplicity, we will change and relax some of the known rules.

2 Definitions

This section provides the basic definitions and the predicate(s) that will be provided to you.

2.1 Catoms

Since the word *atom* is a keyword in SWI-Prolog, we will use the term *catom* as a chemical atom or a ceng atom, whatever you like. A *catom* (Figure 1) has

- some protons and neutrons in its center,
- some electrons orbiting around.

The protons and the neutrons of a catom form its nucleus and *the mass of a catom* is calculated as the sum of the number of protons and the number of neutrons in the nucleus of a catom.

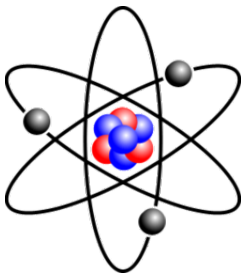

Figure 1: A basic figure of a catom where the red and blue particles represent protons and neutrons in the nucleus and the grey particles represent the electrons orbiting around the nucleus.

Electrons, on the other hand, reside on circular shells, orbiting around the nucleus of a catom where the electrons on the outer-most shell are called *valence* electrons. The distance between shells are equal. Thus, the distance of the closest shell to the nucleus is equal to R/S where the distance of outer-most shell to the nucleus is equal to the radius R of a catom with S number of shells.

In its normal form, a catom has equal number of protons and electrons and the *catomic number* of a catom is defined as the number of protons it has in its nucleus.

A catom tends to exchange electrons to get its outer-most shell with either 0 or 8 electrons, satisfying the octet rule. Here, the other shells of a catom do not have to be full to satisfy the octet rule. For example; a catom with a list of electrons, [5, 3, 4, 6, 2], can loose 2 valence electrons and satisfy the octet rule.

A catom can gain valence electrons to form a negatively charged ion, an *anion*, or loose valence electrons to form a positively charged ion, a *cation*. In our setting, only the catoms with more than 4 valence electrons can gain electrons and form an anion, while the rest of them loose electrons and form a cation. Eventually, anions and cations get together to form an *ionic bonded molecule*.

2.2 `catom/4` predicate

In this assignment, catoms will be given as `catom/4` predicates to you in a Prolog file named as **catoms.pl**.

- A catom is given as;

```
catom(NAME, MASS, RADIUS, ELECTRON_LIST).
```

- `NAME` is a Prolog atom that represents the name of the catom,
- `MASS` is a positive integer that represents the mass of the catom, which is the sum of the number of protons and neutrons in its nucleus,
- `RADIUS` is a positive float that represents the radius of the catom, which is the distance from the outer-most shell to its nucleus,
- `ELECTRON_LIST` is a list that contains the number of electrons in each shell as a positive integer. The order of the list is from the closest to the furthest to the nucleus,
- `NAME` of a catom is unique, i.e. no two `catom/4` predicates will share the same name,
- A catom is guaranteed to have at least 2 shells carrying electrons,
- In our setting, each shell has a capacity of 8 electrons,
- A shell may not be completely filled. For example; [4, 7, 5, 3] is a valid electron configuration in our setting.

For example;

```
catom(iron, 56, 2.20, [2, 8, 8, 6, 2]).
```

represents the catom Iron, with a mass of 56, a radius of 2.20 units. It has 2 electrons in the inner-most shell, 8 in the next one, and so on. It has 2 valence electrons in the outer-most shell, that means that it tends to loose them to form a cation in an ionic bonded molecule. In its normal form, it has a total of 26 electrons and protons while the number of neutrons is equal to 30. An iron catom can give its 2 valence electrons to an Oxygen catom (below) and together they can form an ionic bond. In this Iron Oxide molecule, Iron has 0 valence electrons where Oxygen has 8, satisfying the octet rule. Note that in our setting, the shells before the outer-most shell do not have to be completely filled.

```
catom(oxygen, 16, 1.18, [2, 6]).
```

2.3 Example `catoms.pl` File

```
:- module(catoms, [catom/4]).

% catom(NAME, MASS, RADIUS, ELECTRON_LIST)
catom(carbon, 13, 1.56, [2, 4]).
catom(iron, 56, 2.20, [2, 8, 8, 6, 2]).
catom(bromine, 80, 2.40, [4, 8, 8, 8, 7]).
catom(manganese, 55, 2.58, [7, 8, 8, 2]).
catom(cobalt, 59, 2.10, [5, 8, 9, 6]).
catom(nitrogen, 15, 1.04, [2, 5]).
catom(oxygen, 16, 1.18, [2, 6]).
catom(magnesium, 25, 1.82, [2, 8, 2]).
catom(calcium, 41, 2.01, [2, 8, 8, 2]).
catom(boron, 11, 0.58, [2, 3]).
```

3 Task

This section explains the task and the predicates that you are expected to implement. You should implement three predicates as defined in the following sections.

3.1 `catomic_number/2` predicate (10 Points)

`catomic_number` predicate has 2 arguments representing the name of a catom and the catomic number of it. Again, the catomic number of a catom is the number of protons in its nucleus, which is equal to the total number of electrons in its normal form.

- The predicate is defined as;

```
catomic_number(NAME, CATOMIC_NUMBER).
```

- `NAME` is a Prolog atom that represents a unique name of the catom,
- `CATOMIC_NUMBER` is a positive integer that represents the catomic number of the catom,
- A query missing both `NAME` and `CATOMIC_NUMBER` should find a catom that has the `NAME` and the `CATOMIC_NUMBER` in the knowledge base (`catoms.pl`),
- A query missing `NAME` should find the `NAME` of a catom that has the given `CATOMIC_NUMBER`,
- A query missing `CATOMIC_NUMBER` should find the `CATOMIC_NUMBER` of the catom that has the given `NAME`,
- A query providing both of the arguments should evaluate as `true` if there is a catom with the given `NAME` and `CATOMIC_NUMBER`,
- If there is no matching in the knowledge base (`catoms.pl`), the predicate should evaluate as `false`.

3.1.1 Example Runs

```
?- catomic_number(cobalt, CATOMIC_NUMBER).
CATOMIC_NUMBER = 28.

?- catomic_number(NAME, 12).
NAME = magnesium.

?- catomic_number(NAME, CATOMIC_NUMBER).
NAME = carbon,
CATOMIC_NUMBER = 6 ;
NAME = iron,
CATOMIC_NUMBER = 26 ;
NAME = bromine,
CATOMIC_NUMBER = 35 ;
NAME = manganese,
CATOMIC_NUMBER = 25 ;
NAME = cobalt,
CATOMIC_NUMBER = 28 ;
NAME = nitrogen,
CATOMIC_NUMBER = 7 ;
NAME = oxygen,
CATOMIC_NUMBER = 8 ;
NAME = magnesium,
CATOMIC_NUMBER = 12 ;
NAME = calcium,
CATOMIC_NUMBER = 20 ;
NAME = boron,
CATOMIC_NUMBER = 5.
```

3.2 `ion/2` predicate (20 Points)

`ion` predicate has 2 arguments representing the name of a catom and charge of it when it gets ionized. A catom with more than 4 valence electrons (outer-most shell electrons) gains the remaining electrons and forms an anion while a catom with less than or equal to 4 valence electrons looses those electrons and forms a cation.

- The predicate is defined as;

```
ion(NAME, CHARGE).
```

- `NAME` is a Prolog atom that represents a unique name of the catom,
- `CHARGE` is an integer that represents the charge of the catom,
- A query missing both `NAME` and `CHARGE` should find an ion that has the `NAME` and the `CHARGE`,
- A query missing `NAME` should find the `NAME` of the ion with the given `CHARGE`,
- A query missing `CHARGE` should find the `NAME` of the ion that has the given `NAME`,
- A query providing both of the arguments should evaluate as `true` if there is a ion with the given `NAME` and `CHARGE`,

- If there is no matching, the predicate should evaluate as `false`,
- If the number of valence electrons of a catom is bigger than 4, it gets more electrons to complete its outer-most shell to 8, forming an anion. Otherwise, it looses the electrons and forms a cation.

3.2.1 Example Runs

```
?- ion(boron, CHARGE).
CHARGE = 3 .

?- ion(NAME, 2).
NAME = iron ;
NAME = manganese ;
NAME = magnesium ;
NAME = calcium ;
false.

?- ion(NAME, CHARGE).
NAME = carbon,
CHARGE = 4 ;
NAME = iron,
CHARGE = 2 ;
NAME = manganese,
CHARGE = 2 ;
NAME = magnesium,
CHARGE = 2 ;
NAME = calcium,
CHARGE = 2 ;
NAME = boron,
CHARGE = 3 ;
NAME = bromine,
CHARGE = -1 ;
NAME = cobalt,
CHARGE = -2 ;
NAME = nitrogen,
CHARGE = -3 ;
NAME = oxygen,
CHARGE = -2 ;
false.
```

3.3 `molecule/2` predicate (70 Points)

`molecule` predicate has 2 arguments, one for a list of catom names and one for a total catomic number. The predicate states that the catoms in the list forms an *ionic bonded molecule* with the given total catomic number by exchanging electrons.

- The predicate is defined as;

```
molecule(CATOM_LIST, TOTAL_CATOMIC_NUMBER).
```

- `CATOM_LIST` is a list with the names of catoms **ordered by their catomic numbers**,

- `TOTAL_CATOMIC_NUMBER` is a positive integer which is equal to the sum of the catomic numbers of the catoms in the list,
- A query missing `CATOM_LIST` should find a `CATOM_LIST` where the sum of catomic numbers of the atoms in the list is equal to `TOTAL_CATOMIC_NUMBER` and the sum of the charges of their ions is equal to 0,
- A query providing both of the arguments should evaluate as `true` if the sum of catomic numbers of the atoms in the `CATOM_LIST` is equal to `TOTAL_CATOMIC_NUMBER` and the sum of the charges of their ions is equal to 0. It should evaluate as `false`, otherwise,
- `CATOM_LIST` may have duplicates,
- The catoms in the list may form more of the same molecule. For example; [oxygen, oxygen, magnesium, magnesium] forms two Magnesium Oxide molecules with a total catomic number of 40,
- It is guaranteed that a query of this predicate will be always given with `TOTAL_CATOMIC_NUMBER` argument.

3.3.1 Example Runs

```
?- molecule(CATOM_LIST, 20).
CATOM_LIST = [oxygen, magnesium] .
?- molecule(CATOM_LIST, 30).
false.
?- molecule(CATOM_LIST, 32).
CATOM_LIST = [carbon, nitrogen, nitrogen, magnesium] ;
CATOM_LIST = [boron, nitrogen, oxygen, magnesium] ;
false.
?- molecule(CATOM_LIST, 62).
CATOM_LIST = [oxygen, oxygen, calcium, iron] ;
CATOM_LIST = [boron, nitrogen, nitrogen, nitrogen, magnesium, magnesium, magnesium] ;
CATOM_LIST = [carbon, oxygen, oxygen, oxygen, oxygen, magnesium, magnesium] ;
CATOM_LIST = [carbon, carbon, nitrogen, nitrogen, oxygen, oxygen, calcium] ;
CATOM_LIST = [boron, carbon, nitrogen, oxygen, oxygen, oxygen, calcium] ;
CATOM_LIST = [boron, boron, oxygen, oxygen, oxygen, oxygen, calcium] ;
CATOM_LIST = [carbon, oxygen, oxygen, magnesium, cobalt] ;
CATOM_LIST = [carbon, cobalt, cobalt] ;
false.
?- molecule([oxygen, oxygen, calcium, iron], 62).
true .
```

4 Regulations

1. **Programming Language:** You should write your code using SWI Prolog.
2. **Late Submission:** See the syllabus for the details.
3. **Cheating:** All the work should be done individually. **We have zero tolerance policy for cheating**. People involved in cheating will be punished according to the university regulations.
4. **Cengclass:** You must follow the related Discussion Forum in CengClass (http://cengclass.ceng.metu.edu.tr) for discussions and possible updates on a daily basis.
5. **Evaluation:** Your program will be evaluated automatically using "black-box" technique so make sure to obey the specifications. **Your codes will be tested with a time limit**, so make sure that your code is efficient. For example; in the given knowledge base, finding all possible molecules with total catomic number of 180, should be able to evaluated under 60 seconds. That is, the following query should evaluate without exceeding time limit.

```
?- call_with_time_limit(60, findall(CATOM_LIST, molecule(CATOM_LIST, 180), ALL_CATOM_LIST)).
```

5 Submission

Submission will be done via Ceng Class. Submit a single file called "hw5.pl" through the Ceng Class system. The file **MUST** start with the following lines.

```
:- module(hw5, [catomic_number/2, ion/2, molecule/2]).
:- [catoms].
```