PROLOG

**Knowledge Base 1**

woman(mia).  
   woman(jody).  
   woman(yolanda).  
   playsAirGuitar(jody).  
   party.

 ?-  woman(mia).

Prolog will answer

    yes

?-  playsAirGuitar(jody).

Prolog will again answer yes

?-  playsAirGuitar(mia).

We will get the answer no

?-  playsAirGuitar(vincent).

?-  tatooed(jody).

?-  party.

then Prolog will respond

    yes

and if we pose the query

   ?-  rockConcert.

then Prolog will respond

   no

#### Knowledge Base 2

happy(yolanda).  
   listens2Music(mia).  
   listens2Music(yolanda):-  happy(yolanda).  
   playsAirGuitar(mia):-  listens2Music(mia).  
   playsAirGuitar(yolanda):-  listens2Music(yolanda).

There are two facts in KB2, listens2Music(mia) and happy(yolanda) . The last three items it contains are rules.

?-  playsAirGuitar(mia).

playsAirGuitar(mia):-  listens2Music(mia).

 ?-  playsAirGuitar(yolanda).

listens2Music(yolanda):-  happy(yolanda).

playsAirGuitar(yolanda):-  listens2Music(yolanda).

#### Knowledge Base 3

happy(vincent).  
   listens2Music(butch).  
   playsAirGuitar(vincent):-  
         listens2Music(vincent),  
         happy(vincent).  
   playsAirGuitar(butch):-  
         happy(butch).  
   playsAirGuitar(butch):-  
         listens2Music(butch).

There are two facts, happy(vincent) and listens2Music(butch) , and three rules.

?-  playsAirGuitar(vincent).

?-  playsAirGuitar(butch).

There is another way of expressing disjunction in Prolog. We could replace the pair of rules given above by the single rule

playsAirGuitar(butch):-  
         happy(butch);  
         listens2Music(butch).

#### Knowledge Base 4

woman(mia).  
   woman(jody).  
   woman(yolanda).  
     
   loves(vincent,mia).  
   loves(marsellus,mia).  
   loves(pumpkin,honey\_bunny).  
   loves(honey\_bunny,pumpkin).

?-  woman(X).

X  =  mia  ;  
X  =  jody  ;  
X  =  Yolanda

?-  loves(marsellus,X),  woman(X).

X  =  mia

#### Knowledge Base 5

loves(vincent,mia).  
   loves(marsellus,mia).  
   loves(pumpkin,honey\_bunny).  
   loves(honey\_bunny,pumpkin).  
     
   jealous(X,Y):-  loves(X,Z),  loves(Y,Z).

?-  jealous(marsellus,W).

W  =  Vincent

### **Prolog Syntax**

#### There are four kinds of term in Prolog: atoms, numbers, variables, and complex terms (or structures). Atoms and numbers are lumped together under the heading constants, and constants and variables together make up the simple terms of Prolog.

#### Atoms

An atom is either:

1. A string of characters made up of upper-case letters, lower-case letters, digits, and the underscore character, that begins with a lower-case letter. Here are some examples: butch , big\_kahuna\_burger , listens2Music and playsAirGuitar .
2. An arbitrary sequence of characters enclosed in single quotes. For example ’ Vincent ’, ’ The  Gimp ’, ’ Five\_Dollar\_Shake ’, ’ &^%&#@$  &\* ’, and ’   ’. The sequence of characters between the single quotes is called the atom name. Note that we are allowed to use spaces in such atoms; in fact, a common reason for using single quotes is so we can do precisely that.
3. A string of special characters. Here are some examples: @= and ====> and ; and :- are all atoms. As we have seen, some of these atoms, such as ; and :- have a pre-defined meaning.

#### Numbers

Real numbers aren’t particularly important in typical Prolog applications. So although most Prolog implementations do support floating point numbers or floats (that is, representations of real numbers such as 1657.3087 or π ) we say little about them in this book.But integers (that is: …,-2, -1, 0, 1, 2, 3,…) are useful for such tasks as counting the elements of a list,. Their Prolog syntax is the obvious one: 23 , 1001 , 0 , -365 , and so on.

#### Variables

A variable is a string of upper-case letters, lower-case letters, digits and underscore characters that starts either with an upper-case letter or with an underscore.

For example, X , Y , Variable , \_tag , X\_526 , List , List24 , \_head , Tail , \_input and Output are all Prolog variables.

The variable \_ (that is, a single underscore character) is rather special. It’s called the anonymous variable

#### Complex terms

Constants, numbers, and variables are the building blocks: now we need to know how to fit them together to make complex terms. Recall that complex terms are often called structures.

Complex terms are build out of a functor followed by a sequence of arguments. The arguments are put in ordinary parentheses, separated by commas, and placed after the functor. Note that the functor has to be directly followed by the parenthesis; you can’t have a space between the functor and the parenthesis enclosing the arguments. The functor must be an atom. That is, variables cannot be used as functors. On the other hand, arguments can be any kind of term.

Now, we’ve already seen lots of examples of complex terms when we looked at the knowledge bases KB1 to KB5. For example, playsAirGuitar(jody) is a complex term: its functor is playsAirGuitar and its argument is jody . Other examples are loves(vincent,mia) and, to give an example containing a variable, jealous(marsellus,W) .

### **Exercises**

**Exercise**  1.1 Which of the following sequences of characters are atoms, which are variables, and which are neither?

1. vINCENT
2. Footmassage
3. variable23
4. Variable2000
5. big\_kahuna\_burger
6. ’big  kahuna  burger’
7. big  kahuna  burger
8. ’Jules’
9. \_Jules
10. ’\_Jules’

**Exercise**  1.2 Which of the following sequences of characters are atoms, which are variables, which are complex terms, and which are not terms at all? Give the functor and arity of each complex term.

1. loves(Vincent,mia)
2. ’loves(Vincent,mia)’
3. Butch(boxer)
4. boxer(Butch)
5. and(big(burger),kahuna(burger))
6. and(big(X),kahuna(X))
7. \_and(big(X),kahuna(X))
8. (Butch  kills  Vincent)
9. kills(Butch  Vincent)
10. kills(Butch,Vincent

**Exercise**  1.3 How many facts, rules, clauses, and predicates are there in the following knowledge base? What are the heads of the rules, and what are the goals they contain?

   woman(vincent).  
   woman(mia).  
   man(jules).  
   person(X):-  man(X);  woman(X).  
   loves(X,Y):-  father(X,Y).  
   father(Y,Z):-  man(Y),  son(Z,Y).  
   father(Y,Z):-  man(Y),  daughter(Z,Y).

**Exercise**  1.4 Represent the following in Prolog:

1. Butch is a killer.
2. Mia and Marsellus are married.
3. Zed is dead.
4. Marsellus kills everyone who gives Mia a footmassage.
5. Mia loves everyone who is a good dancer.
6. Jules eats anything that is nutritious or tasty.

**Exercise**  1.5 Suppose we are working with the following knowledge base:

   wizard(ron).  
   hasWand(harry).  
   quidditchPlayer(harry).  
   wizard(X):-  hasBroom(X),  hasWand(X).  
   hasBroom(X):-  quidditchPlayer(X).

How does Prolog respond to the following queries?

1. wizard(ron).
2. witch(ron).
3. wizard(hermione).
4. witch(hermione).
5. wizard(harry).
6. wizard(Y).
7. witch(Y).

%  w-mother-in-law(X,Y) means X is Y’s wife’s mother.  
%  w-father-in-law(X,Y) means X is Y’s wife’s father.  
father(vin,aby).  
father(ral,nic).  
father(chris,vin).  
father(vin,sky).  
mother(gina,vin).  
mother(nic,aby).  
mother(ala,nic).  
mother(nic,sky).  
female(aby).  
female(nic).  
female(ala).  
female(gina).  
female(sky).  
male(ral).  
male(vin).  
male(chris).  
parent(X,Y):-father(X,Y);mother(X,Y).  
grandparent(X,Y):-parent(X,M),parent(M,Y).  
children(X,Y):-parent(Y,X).  
husband(X,Y):-male(X),female(Y),children(M,X),children(M,Y).  
sibling(X,Y):-mother(M,X),mother(M,Y),father(F,X),father(F,Y),X=Y.  
grandfather(X,Y):-father(X,K),father(K,Y).  
grandfather(X,Y):-father(X,K),mother(K,Y).  
grandmother(X,Y):-mother(X,K),mother(K,Y).  
grandmother(X,Y):-mother(X,K),father(K,Y).  
h-mother-in-law(X,Y):-mother(X,K),male(K),husband(K,Y).  
w-mother-in-law(X,Y):-mother(X,K),female(K),husband(Y,K).





