

Chapter 1

Chapter 1 exercises

1.1 Exercise 1.1

- (a) Which of the following are valid Prolog atoms?

This was verified in SWI-Prolog using `atom()`

`f`: Valid
`loves(john, mary)`: Invalid, this is a compound term
`Mary`: Invalid, this is a variable
`_c1`: Invalid, this is a variable
`'Hello'`: Valid, you can use single quotes to make any string an atom
`this_is_it`: Valid

- (b) Which of the following are valid names for Prolog variables?

This was verified in SWI-Prolog using `var()`

`a`: Invalid, must start with uppercase or underscore
`A`: Valid
`Paul`: Valid
`'Hello'`: Invalid, single quotes denote atoms
`a_123`: Invalid
`_`: Valid, anonymous variable
`_abc`: Valid
`x2`: Valid

- (c) What would a Prolog interpreter reply given the following query?

`?- f(a, b) = f(X, Y).`

The interpreter would instantiate `X = a` and `Y = b`, as this will evaluate the query to be true:

`X = a`
`Y = b`
`true`

- (d) Would the following query succeed?

`?- loves(mary, john) = loves(John, Mary).`

This query would succeed. Both John and Mary are variables, so the interpreter would instantiate `John = john` and `Mary = mary`, causing the statements to match.

- (e) Assume a program consisting only of the fact `a(B, B).` has been consulted by Prolog. How will the system react to the following query?

`?- a(1, X), a(X, Y), a(Y, Z), a(Z, 100).`

Prolog will return **false**. The first query causes `X = 1`, the subsequent query causes `Y = 1`, then `Z = 1`, and the final query will fail as 1 and 100 are different.

1.2 Exercise 1.2

Understand and explain the following queries:

- (a) `?- myFunctor(1, 2) = X, X = myFunctor(Y, Y).`

The interpreter will instantiate $X = \text{myFunctor}(1, 2)$ to match the first clause. On the following clause, the equality fails, as the numbers inside the compound term X are not the same. As Y cannot be instantiated to both 1 and 2, the query is **false**.

- (b) `?- f(a, _, c, d) = f(a, X, Y, _).`

The interpreter will match the `_` on the left-hand side to X . It will then instantiate $Y = c$, and the `_` on the right-hand side to d . The query will return **true**. An interesting thing to note from running the query is that only the $Y = c$ instantiation is displayed, as the `_` variables are ignorable.

- (c) `?- write('One '), X = write('Two ').`

The `write()` predicate always returns true (as well as writing text in the command line). This means the first clause passes, and the string literal "One " is printed. For the second clause, the variable X is initialised as $X = \text{write}('Two')$, meaning the interpreter instead views `write()` as a compound term that it can assign to X . This makes the equality pass and so the overall query returns **true**.

1.3 Exercise 1.3

Draw the family tree corresponding to the following Prolog program:

N.B. I couldn't be bothered writing out everything, for reference see Exercise 1.3 of [PrologPrimer.pdf](#)

