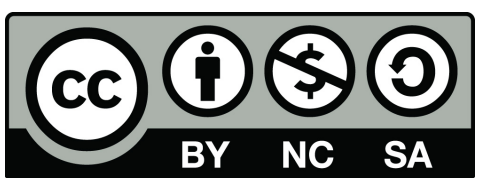
Inteligencia Artificial: Prolog

Listas

Universidad de La Laguna

Christopher Expósito-Izquierdo[1],
Belén Melián-Batista[2]

{cexposit[1], mbmelian[2]}@ull.es
Universidad de La Laguna (España)



Contenidos

- ¿Qué es una lista?
- Operador |
- El predicado `member`
- El predicado `deleteAll`
- El predicado `append`
- Librería `lists.pl`

¿Qué es una lista?

Una lista en Prolog es una estructura de datos que permite almacenar cualquier tipo de elemento definido en el lenguaje: constantes, variables y estructuras

- Las listas se definen encerrando sus elementos entre corchetes
- Los elementos de una lista se separan por comas
- La longitud de una lista es el número de elementos que contiene
- Una lista puede contener un número finito de elementos
- Los elementos de una lista pueden ser, a su vez, listas
- Existe una lista especial denominada 'lista vacía' ([]) cuya longitud es cero

¿Qué es una lista? Ejemplos

- `[antonio, luis, andrea, marta, pepe]`
- `[antonio, Z, 14, correr(maria), antonio]`
- `[antonio, saltar(pepe), [14, K, maria], luis]`
- `[]`
- `[[pepe,[luis, [maria]]], X, [], 92, correr(luis), []]`

Operador |

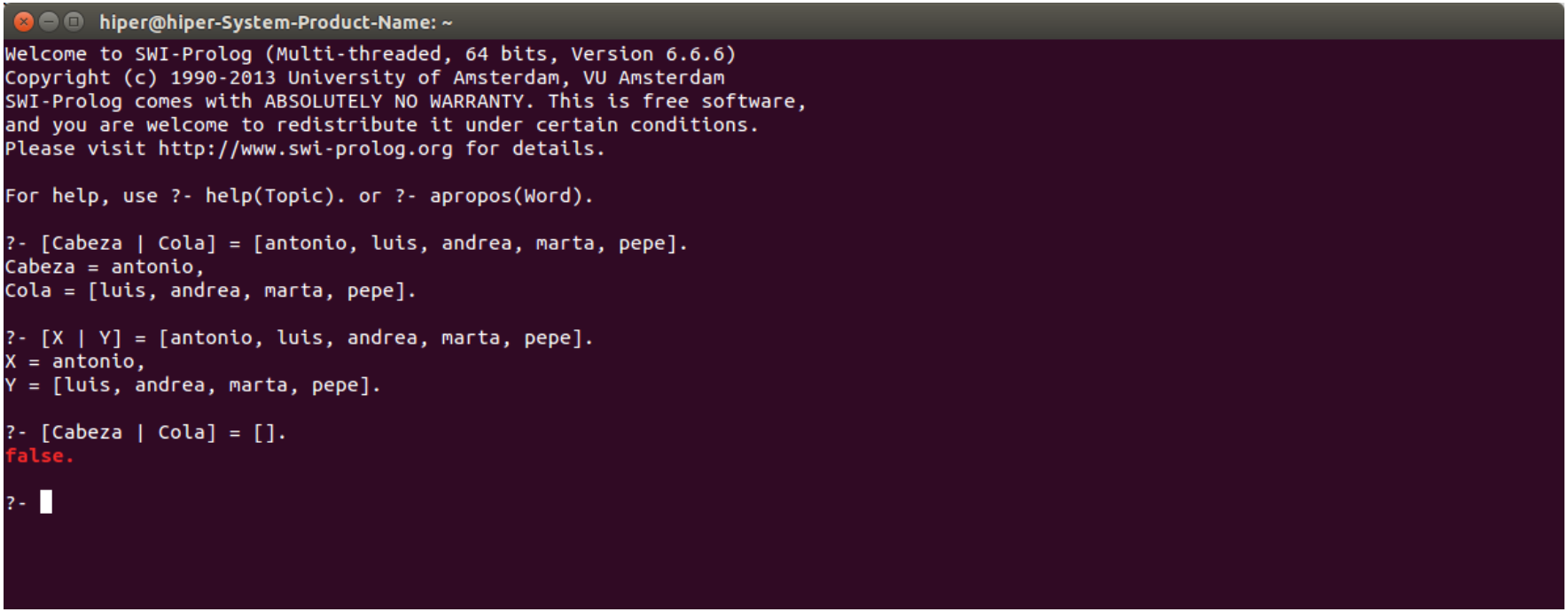
- Las listas no vacías en Prolog pueden descomponerse en cabeza y cola. Sin embargo, una lista vacía por definición no puede descomponerse de esta manera

- La cabeza y la cola de una lista se definen como sigue:
  - La cabeza de una lista es su primer elemento
  - La cola de una lista es una lista compuesta por todos los elementos que no son la cabeza

- El operador | es posiblemente el operador más importante en Prolog para la manipulación de listas de elementos. Está destinado a realizar la descomposición de una lista en su cabeza y su cola

Operador |

A continuación podemos ver varios ejemplos del uso del operador |:



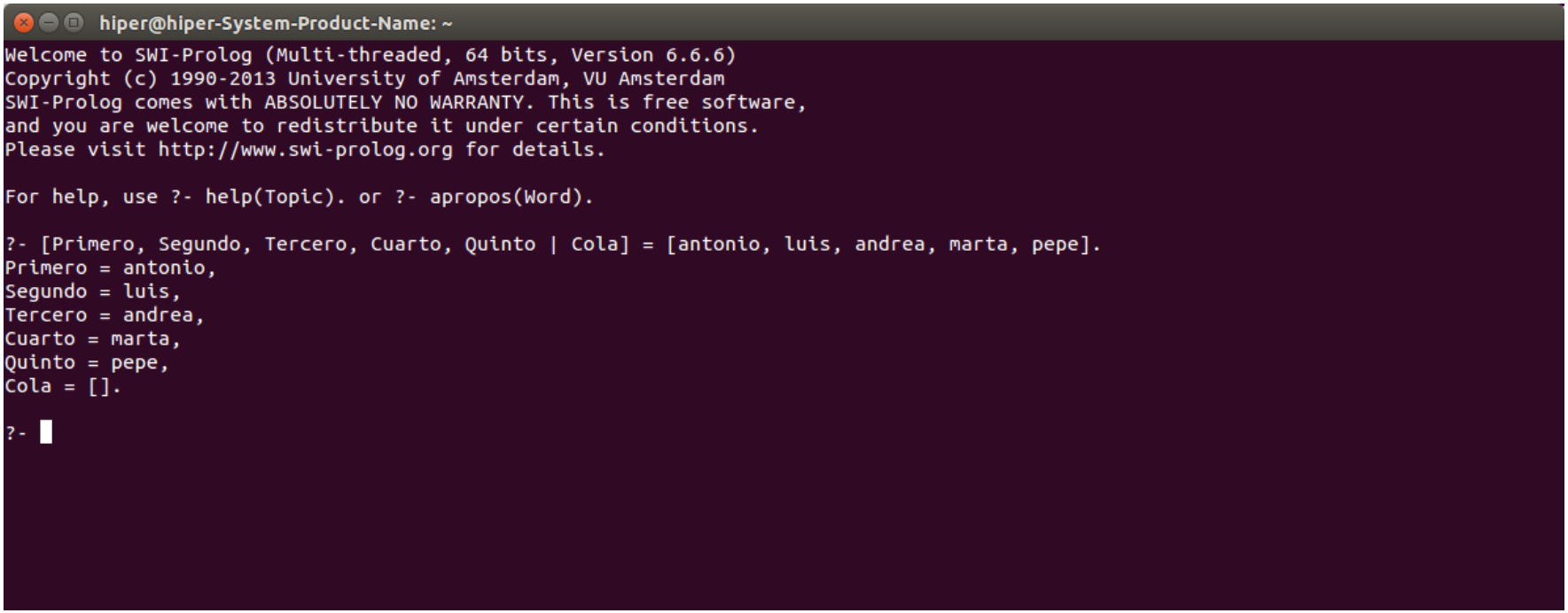
Operador |

El operador | permite obtener los primeros *n* elementos de una lista. Para ello hay que especificar *n* variables a la izquierda del operador. A continuación se muestran ejemplos en los que se extraen 1, 2 y 3 primeros elementos de una lista:

```
hiper@hiper-System-Product-Name: ~
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.6.6)
Copyright (c) 1990-2013 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- [Primero | Cola] = [antonio, luis, andrea, marta, pepe].
Primero = antonio,
Cola = [luis, andrea, marta, pepe].

?- [Primero, Segundo | Cola] = [antonio, luis, andrea, marta, pepe].
Primero = antonio,
Segundo = luis,
Cola = [andrea, marta, pepe].

?- [Primero, Segundo, Tercero | Cola] = [antonio, luis, andrea, marta, pepe].
Primero = antonio,
Segundo = luis,
Tercero = andrea,
Cola = [marta, pepe].

?-
```

Operador |

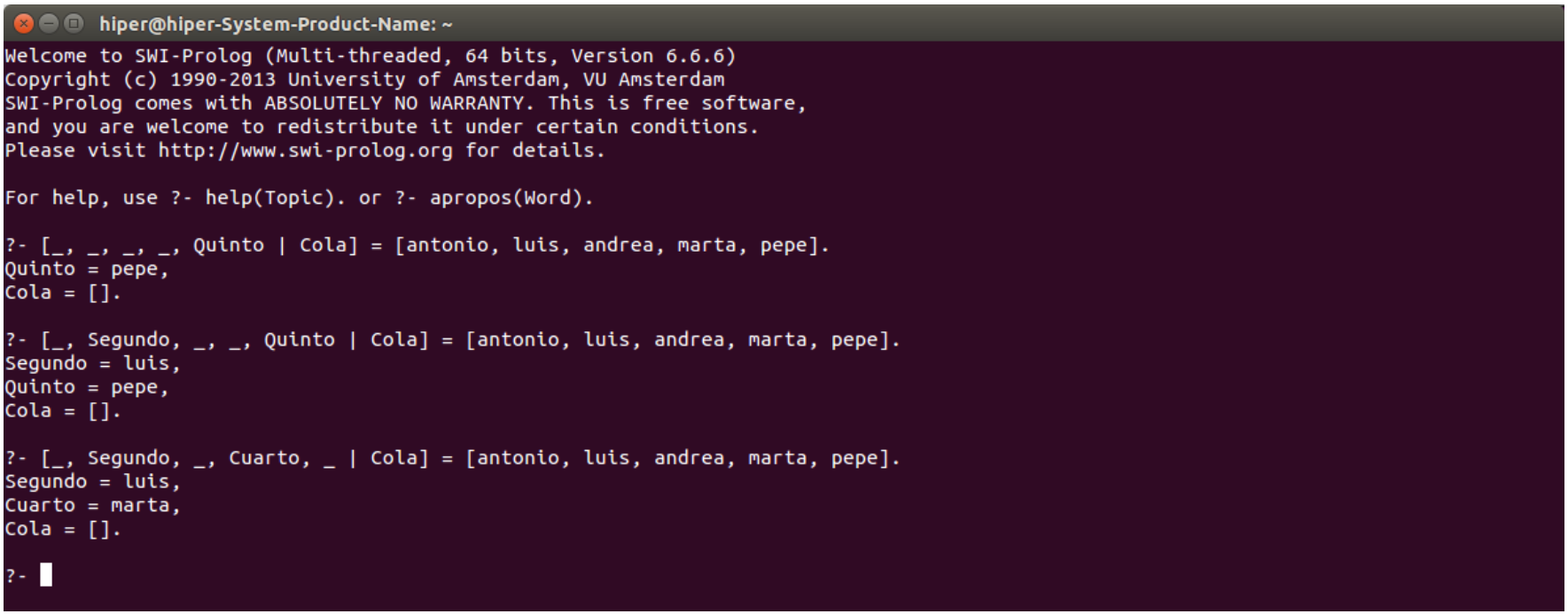
¿Qué sucede si solo nos interesa el quinto elemento de una lista?



Efectivamente, mediante la consulta anterior hemos obtenido el quinto elemento de la lista, tal como deseábamos. Sin embargo, tenemos mucha más información adicional e irrelevante para nosotros

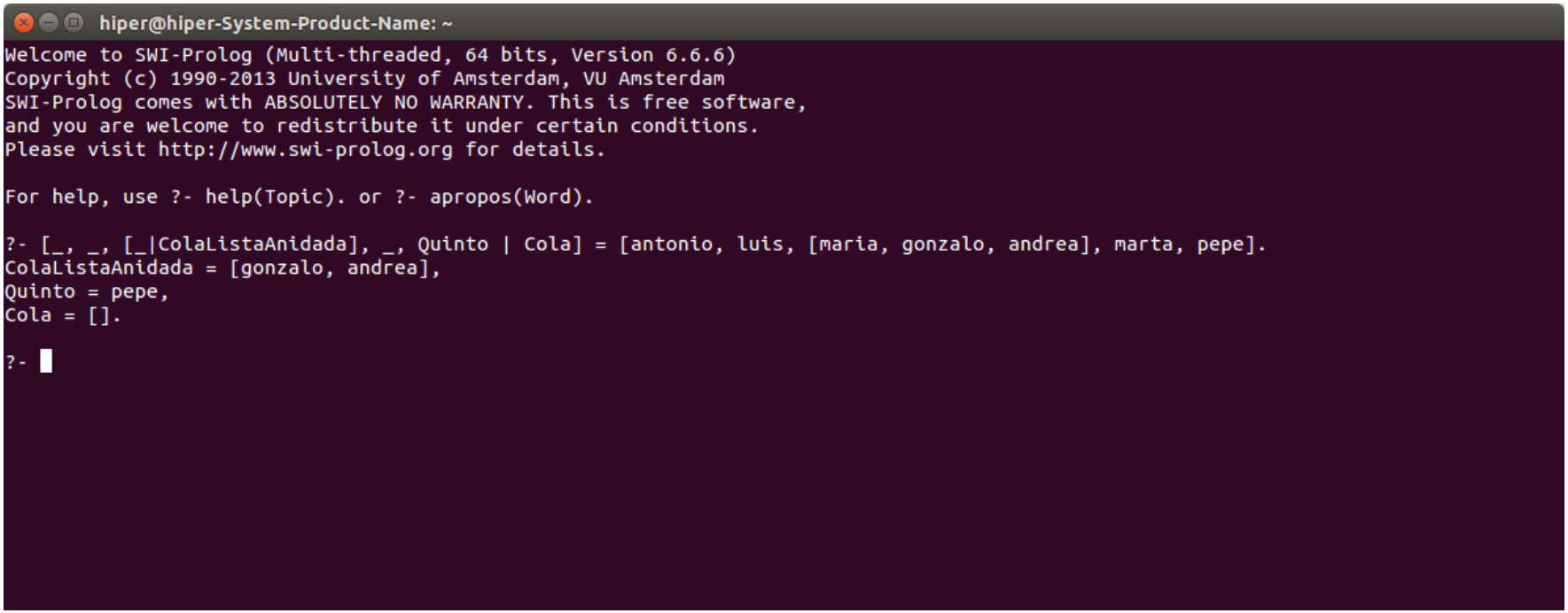
Operador |

En aquellos casos en que no se esté interesado en información adicional a nuestra consulta, se puede evitar que el intérprete nos la proporcione empleando la variable anónima: '_'. A continuación podemos ver un ejemplo de su uso:



Operador |

De forma similar, pueden extraerse únicamente elementos contenidos en listas anidadas. A continuación podemos ver un ejemplo de esto:



El predicado `member`

Objetivo: definir un predicado en Prolog que determine si un cierto elemento se encuentra en una lista

El predicado propuesto debe tener dos argumentos:

1. El elemento a buscar
2. La lista en la que buscar

El predicado `member`

Un elemento está en una lista si es el primer elemento de la misma. Esto significa que el elemento a buscar coincide con la cabeza de la lista.
Se puede expresar en Prolog con el siguiente hecho:

```
member(Elemento, [Elemento | Cola]).
```

El predicado `member`

Por otro lado, un elemento está en una lista en la que no es su cabeza si está en la cola de la misma. Dado que la cola de una lista es, a su vez, una lista, podemos emplear el predicado `member` de forma recursiva para comprobarlo. Esto puede expresarse a través de la siguiente regla:

```
member(Elemento, [Cabeza | Cola]) :-
    member(Elemento, Cola).
```

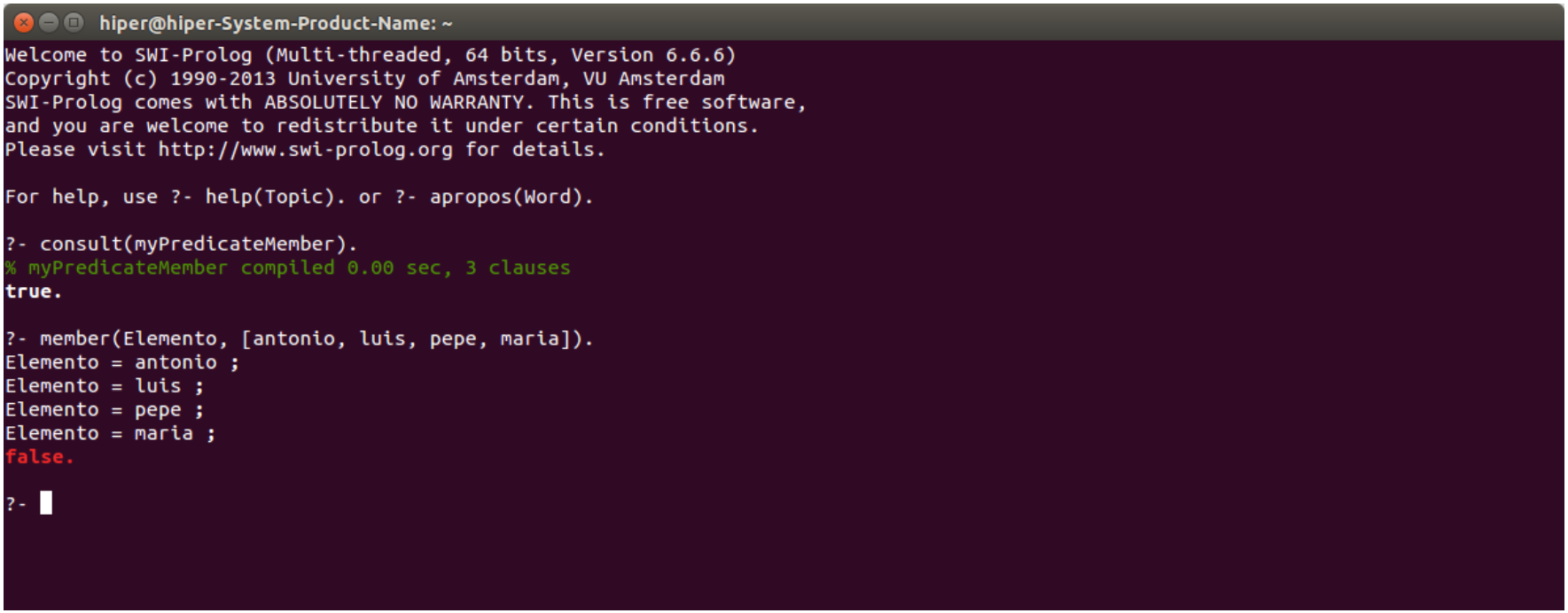
El predicado `member`

La definición del predicado `member` la guardamos en un fichero de texto denominado `myPredicateMember.pl`. Este fichero debe ser cargado en el intérprete antes de poder emplear el predicado. A continuación podemos ver ejemplos del uso:

```
hiper@hiper-System-Product-Name: ~
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.6.6)
Copyright (c) 1990-2013 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- consult(myPredicateMember).
% myPredicateMember compiled 0.00 sec, 3 clauses
true.

?- member(antonio, [antonio, luis, pepe, maria]).
true .

?- member(luis, [antonio, luis, pepe, maria]).
true .

?- member(ricardo, [antonio, luis, pepe, maria]).
false.

?-
```

El predicado `member`

Podemos también consultar cuáles son los elementos miembros de una determinada lista. A continuación vemos un ejemplo:



El predicado `deleteAll`

Objetivo: definir un predicado en Prolog que elimine un cierto elemento de una lista tantas veces como aparezca

El predicado propuesto debe tener tres argumentos:

1. La lista en la que queremos eliminar
2. El elemento a eliminar
3. La lista obtenida al haber eliminado el elemento

El predicado `deleteAll`

Si la lista se encuentra vacía, cualquier elemento que se quiera eliminar da como resultado otra lista vacía. Esto se puede expresar en Prolog con el siguiente hecho:

```
deleteAll([], _, []).
```

El predicado `deleteAll`

Por otro lado, si el elemento a eliminar es el primero de la lista, la lista resultante no se ve afectada en forma alguna. Además, en tal caso, se procede a eliminar el elemento de la cola de la lista. Esto puede expresarse a través de la siguiente regla:

```
deleteAll([Cabeza | Cola], Elemento, Resultado)
                :- Cabeza = Elemento,
      deleteAll(Cola, Elemento, Resultado).
```

El predicado `deleteAll`

Por último, si el elemento a eliminar no es el primero de la lista, la lista resultante debe tener como cabeza ese mismo elemento dado que no va a eliminarse. Además, al igual que en la regla anterior, debemos eliminar el elemento de la cola de la lista. Esto puede expresarse a través de la siguiente regla:

```
deleteAll([Cabeza|Cola],Elemento,[Cabeza|Resultado])
            :- dif(Cabeza, Elemento),
          deleteAll(Cola,Elemento,Resultado)
```

El predicado `deleteAll`

La definición del predicado `deleteAll` la guardamos en un fichero de texto denominado `myPredicateDeleteAll.pl`. Este fichero debe ser cargado en el intérprete antes de poder emplear el predicado. A continuación podemos ver ejemplos del uso:

```
hiper@hiper-System-Product-Name: ~
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.6.6)
Copyright (c) 1990-2013 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- consult(myPredicateDeleteAll).
% myPredicateDeleteAll compiled 0.00 sec, 4 clauses
true.

?- deleteAll([a, b, c, c, b, a], a, Resultado).
Resultado = [b, c, c, b] .

?- deleteAll([a, b, c, c, b, a], b, Resultado).
Resultado = [a, c, c, a] .

?- deleteAll([a, b, c, c, b, a], pepe, Resultado).
Resultado = [a, b, c, c, b, a].

?-
```

El predicado `append`

Objetivo: definir un predicado en Prolog que concatene dos listas de elementos

El predicado propuesto debe tener tres argumentos:

1. La primera de las listas a cuyos elementos van a concatenarse los de la siguiente lista
2. La segunda de las listas cuyos elemento van a ser concatenados a los de la anterior lista
3. La lista obtenida al haber concatenado los elementos de las dos listas anteriores

El predicado `append`

Si la primera de las listas se encuentra vacía, la lista resultante es la misma que la segunda de las listas. Esto puede expresarse a través del siguiente hecho:

```
append([], Lista, Lista).
```

El predicado `append`

Si la primera lista no se encuentra vacía (tiene cabeza y cola), la lista resultante tiene como cabeza la cabeza de la primera lista y como cola la concatenación de los elementos que integran la cola de la primera lista y los de la segunda lista. Esto puede expresarse a través de la siguiente regla:

```
append([Cabeza | Cola], Lista,[Cabeza |
        ColaResultado]) :-
  append(Cola, Lista, ColaResultado).
```

El predicado `append`

La definición del predicado `append` la guardamos en un fichero de texto denominado `myPredicateAppend.pl`. Este fichero debe ser cargado en el intérprete antes de poder emplear el predicado. A continuación podemos ver ejemplos del uso:

```
hiper@hiper-System-Product-Name: ~
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 6.6.6)
Copyright (c) 1990-2013 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- consult(myPredicateAppend).
% myPredicateAppend compiled 0.00 sec, 3 clauses
true.

?- append([], [a, b], Resultado).
Resultado = [a, b].

?- append([c, d, e], [a, b], Resultado).
Resultado = [c, d, e, a, b].

?- append([c, d, e], [], Resultado).
Resultado = [c, d, e].

?-
```

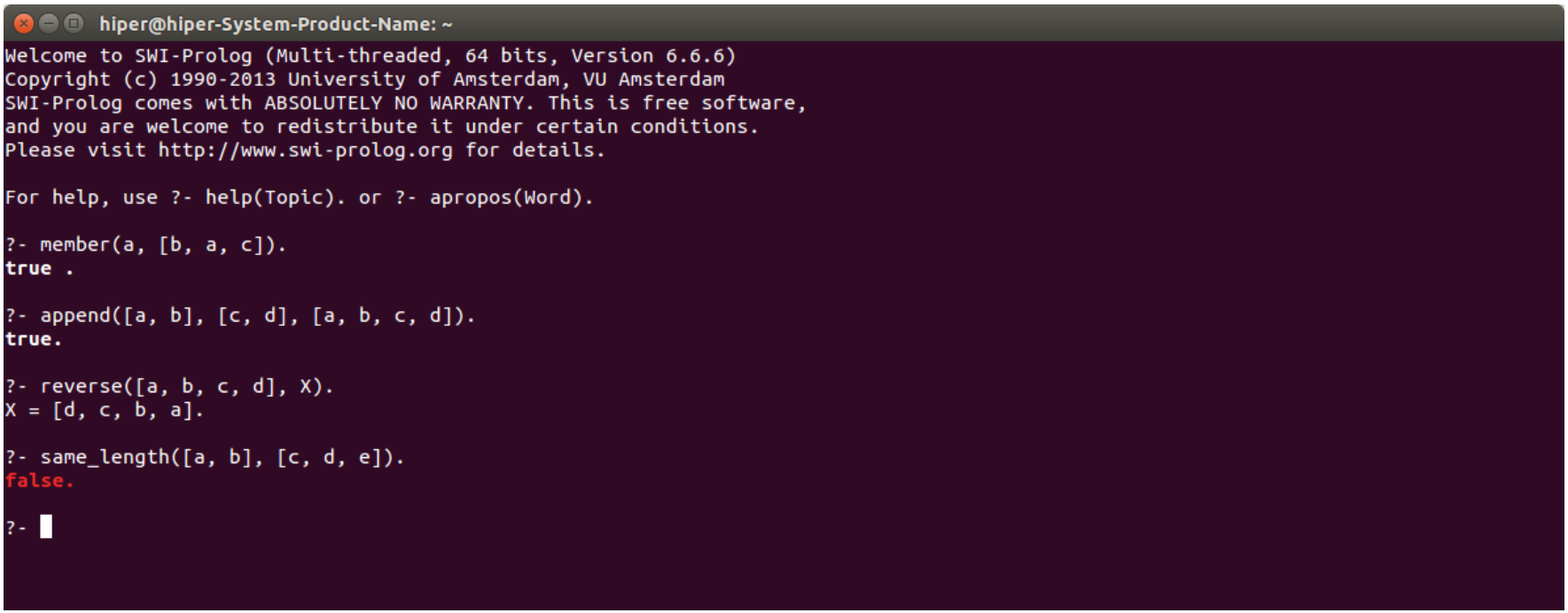
Librería `lists.pl`

Esta librería proporciona un amplio conjunto de predicados destinados a la manipulación de listas en Prolog

- Algunos de los predicados definidos en la librería son: `member`, `append`, `select`, `delete`, `reverse`, `subset`, `union`, `same_length`, etc.
- La librería se encuentra cargada por defecto en SWI Prolog. De esta forma, los predicados definidos en ella pueden ser empleados sin necesidad de realizar ninguna operación
- Documentación acerca de la librería: `http://www.swi-prolog.org/pldoc/man?section=lists`

Librería `lists.pl`

A continuación podemos ver un ejemplo de su uso:



Debe notarse aquí que no se realizado la carga de ningún fichero de predicados para poder hacer uso de la librería.

Inteligencia Artificial: Prolog

Listas

Christopher Expósito-Izquierdo[1],
Belén Melián-Batista[2]

{cexposit[1], mbmelian[2]}@ull.es
Universidad de La Laguna (España)