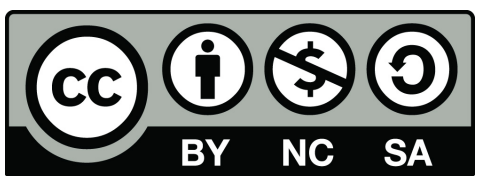
Inteligencia Artificial: Prolog

Introducción al Lenguaje de Programación Prolog

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

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



Contenidos

¿Qué es Prolog?

Prolog (PROgrammation en LOGique) es un lenguaje de programación de propósito general, para computación simbólica y basado en el paradigma de programación denominado programación lógica

Prolog tiene un conjunto de características que deben ser resaltadas:

- Es *declarativo*: se debe especificar cómo es el problema que se desea resolver y cómo son las soluciones a dicho problema, pero no se indica cómo debe alcanzarse dichas soluciones
- Está basado en el concepto de recursión por lo que los bucles no son habitualmente empleados
- Emplea relaciones entre elementos del lenguaje para modelar el entorno
- Usa la unificación

¿Qué es Prolog?

Prolog se emplea en un amplio conjunto de campos. Algunos de ellos son los siguientes:

- Análisis de lenguaje natural:
  `http://mtome.com/Publications/PNLA/pnla.html`
- Demostración de teoremas: A prolog technology theorem prover: Implementation by an extended prolog compiler. Journal of Automated Reasoning. Volumen 4, Issue 4, páginas 353-380 (1988)
  - Autor: M. E. Stickel
- Sistemas expertos: The design of expert systems using Prolog. Prolog and its applications. Springer, páginas 257-271 (1991)
  - Autor: F. Mizoguchi

Historia de Prolog

- Los orígenes de Prolog se remontan al final de la década de los 60 y comienzo de los 70. Sin embargo, no surgió con la intención de ser un lenguaje de programación sino con la intención de poder procesar lenguaje natural, en particular francés.
- Una primera versión del lenguaje apareció al final de 1971 mientras que la primera versión estable fue presentada al final de 1972 en Marsella (Francia) por Alain Colmerauer y Philippe Roussel.
- En sus orígenes, Prolog podía entenderse como el resultado de la unión del procesamiento de lenguaje natural y la demostración automática de teoremas. Sin embargo, su gran variedad de campos de aplicación hace que este enfoque se quede ciertamente limitado.

Historia de Prolog

- En 1977, David Warren presentó el primer compilador de Prolog: DEC-10. Posteriomente, Warren desarrolló la denominada Warren Abstract Machine (WAM), una máquina abstracta destinada a la ejecución de Prolog basada en un conjunto de instrucciones y una arquitectura de memoria. Su relevancia la ha convertido en el estándar *de facto* para los compiladores de Prolog

- Desde la presentación de WAM, la popularidad de Prolog ha crecido incesantemente. Algunos ejemplos de interés por el lenguaje se reflejan en los proyectos de interfaces de procesamiento de lenguaje natural de la NASA para la Estación Espacial Internacional en 2005 o la implementación del ordenador Watson de IBM en 2011

Intérpretes de Prolog

- Amzi! Prolog
  - `http://www.amzi.com/products/prolog_products.htm`
- B-Prolog
  - `http://www.probp.com/`
- Ciao Prolog
  - `http://clip.dia.fi.upm.es/Software/Ciao/`
- GNU Prolog
  - `http://gnu-prolog.inria.fr/`
- Logic Programming Associates Prolog
  - `http://www.lpa.co.uk`
- PD Prolog
  - `http://www-2.cs.cmu.edu/afs/cs/project/ai-repository/ai/lang/prolog/impl/prolog/pdprolog/0.html`

Intérpretes de Prolog

- SICStus Prolog
  - `http://www.sics.se/isl/sicstuswww/site/index.html`
- SWI Prolog
  - `http://www.swi-prolog.org/`
- Turbo Prolog
  - `http://www.fraber.de/university/prolog/tprolog.html`
- Visual Prolog
  - `http://www.visual-prolog.com/`
- W-Prolog
  - `http://waitaki.otago.ac.nz/~michael/wp/`
- YAP Prolog
  - `http://www.ncc.up.pt/~vsc/Yap/`

SWI-Prolog

SWI-Prolog es el intérprete que se va a emplear a lo largo del presente curso debido a que se trata de un intérprete de Prolog libre, robusto y de fácil manejo que posibilita el desarrollo de aplicaciones en un amplio rango de dominios.

La instalación de SWI-Prolog en Ubuntu se realiza mediante la ejecución de los siguientes comandos en la terminal:

1. `sudo apt-add-repository ppa:swi-prolog/stable`
2. `sudo apt-get update`
3. `sudo apt-get install swi-prolog`

Una vez SWI-Prolog ha sido instalado, puede accederse al mismo mediante el siguiente comando: `swipl`

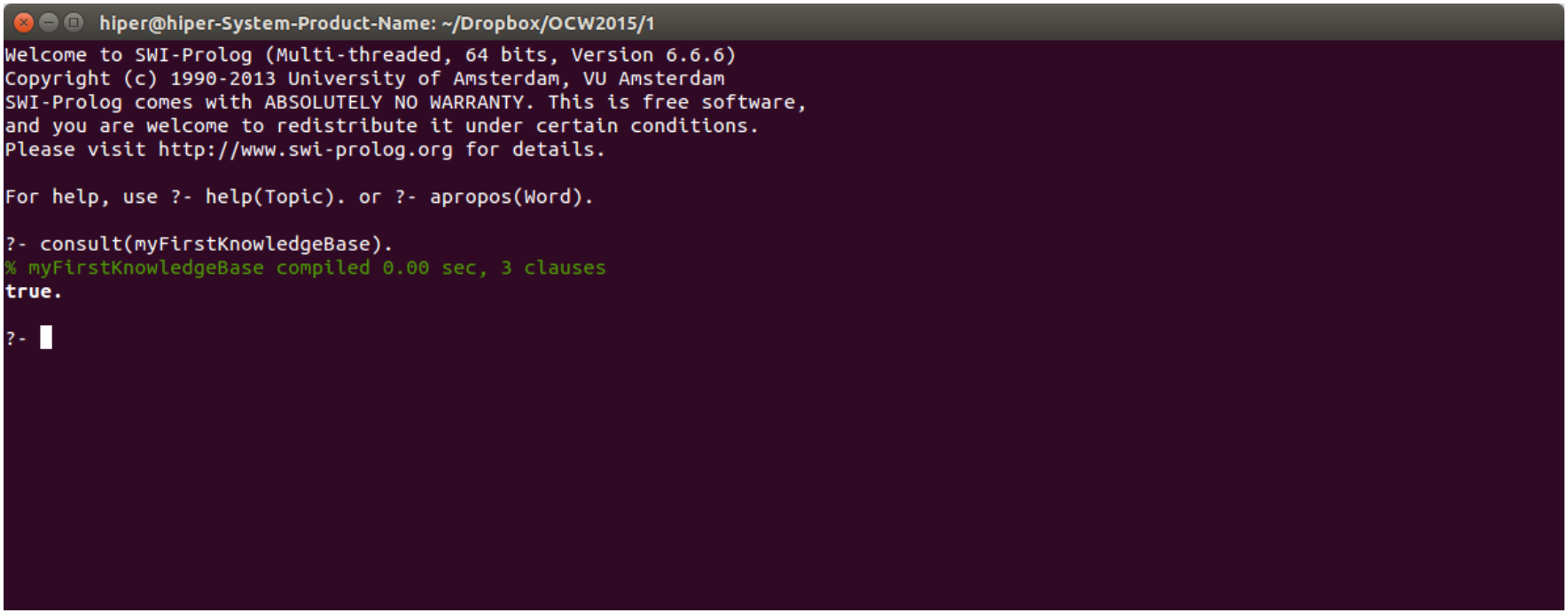
SWI-Prolog

Para salir del intérprete hay que introducir `halt.`

Para más información referente a SWI-Prolog puede consultarse `http://www.swi-prolog.org/pldoc/doc_for?object=manual`

SWI-Prolog

Los programas en Prolog son escritos en ficheros de texto con extensión `.pl` y cargados como podemos ver a continuación:



Para más información puede consultarse

`http://www.swi-prolog.org/pldoc/man?predicate=consult/1`

Átomos, Números, Variables y Términos Complejos

Átomos:

- Son términos compuestos por letras tanto en mayúsculas como en minúsculas, números y _. Además deben comenzar por una letra en minúscula. Ejemplos: `antonio`, `luis27`, `maria_84`, etc.
- Los términos encerrados entre comillas simples también son considerados átomos. Ejemplos: `'Doctor Antonio'`, `'* + )'`, etc.
- Finalmente, varios símbolos válidos en Prolog son considerados como átomos. Ejemplos: +, -, @, etc.
- Algunos átomos, tales como :- tienen un significado predefinido en Prolog

Átomos, Números, Variables y Términos Complejos

Números:

- Prolog define números enteros (por ejemplo: 1, 23, -4, 2119, etc.).
- Algunos intérpretes permiten la definición de números reales (por ejemplo: -27.41, 0.29, 8.641, etc.).
- El rango de valores válidos depende del número de bits empleado para representar dicho número y, por tanto, depende del ordenador empleado y la configuración del intérprete usado.

Átomos, Números, Variables y Términos Complejos

Variables:

- Son términos compuestos por letras, números y _. Además deben comenzar por una letra en mayúscula o con _. Ejemplos: `Antonio`, `Luis27`, `_Maria84`, etc.

- Prolog define un elemento como variable anónima (_) que es similar a cualquier otra variable pero que se emplea para ocupar cualquier posición donde algo es esperado. Cada aparición de la variable anónima denota a una variable distinta

Átomos, Números, Variables y Términos Complejos

Términos complejos:

- Los átomos, números y variables son los elementos básicos definidos en Prolog que permiten la definición de términos complejos. Estos términos complejos son también habitualmente denominados como *estructuras*.
- Un término complejo es un *funtor* acompañado por una secuencia de *argumentos*. Estos argumentos son especificados entre paréntesis, separados por comas y situados inmediatamente detrás del funtor.
- Los funtores deben ser átomos mientras que los argumentos pueden ser cualquier tipo de término, incluidos términos complejos.

Átomos, Números, Variables y Términos Complejos

Ejemplos de términos complejos:

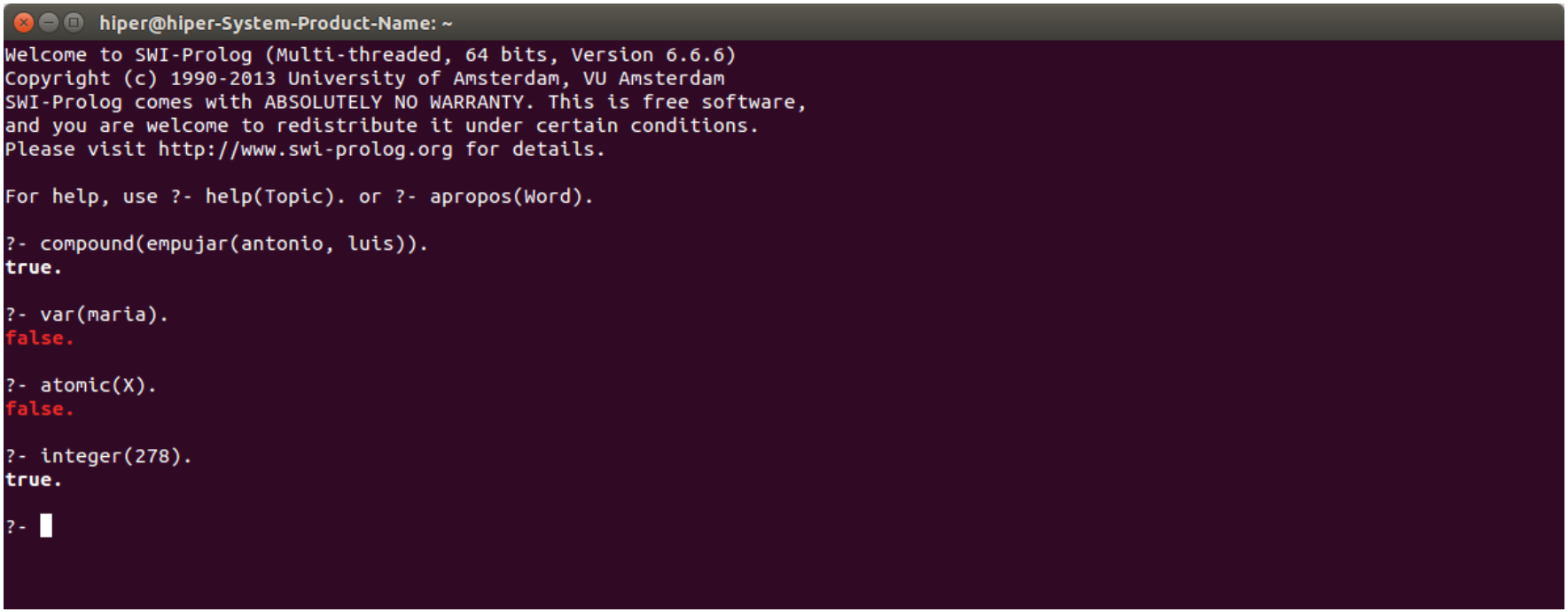
- `miFuntor(Argumento1, argumento_2, Argumento3).`
- `empujar(antonio, luis).`
- `saltar(X).`
- `hablar(X, maria).`
- `grupo(X, maria, Y, luis).`
- `debatir(_, maria, Antonio, grupo(A, maria, pepe, luis)).`

Átomos, Números, Variables y Términos Complejos

Prolog proporciona varios tests que tienen éxito si un término `X` es

- Una variable no instanciada: `var(X)`
- No es una variable o una variable instanciada: `nonvar(X)`
- Un término complejo: `compound(X)`
- Un átomo: `atom(X)`
- Un entero: `integer(X)`
- Un número real: `float(X)`
- Un número real o entero: `number(X)`
- Un número, cadena de texto o átomo: `atomic(X)`

Átomos, Números, Variables y Términos Complejos



Programas en Prolog

La construcción de programas en Prolog se realiza traduciendo los conceptos existentes en lenguaje natural en un lenguaje de primer orden.

Los programas en Prolog están compuestos por un conjunto de claúsulas. Las claúsulas pueden ser de tres tipos:

- *Hechos*: declaran cosas que son siempre ciertas
- *Reglas*: declaran cosas que son condicionalmente ciertas
- *Consultas*: preguntas realizadas por parte del usuario acerca de qué cosas son ciertas

Las bases de conocimiento están compuestas por hechos y reglas.

Programas en Prolog: Hechos

Los hechos son declaraciones siempre verdaderas. Éstos pueden ser elementos individuales o relaciones entre elementos. A continuación se pueden ver algunos ejemplos de hechos:

- `antonio.`
- `saltar(andrea, muro).`
- `correr(sandra).`
- `padre(luis, carol).`
- `marta.`

Para introducir los hechos en el intérprete, los escribimos en un fichero de texto: `myFacts.pl`

Programas en Prolog: Reglas

- Las reglas en Prolog son sentencias con la siguiente forma:

$$A :- \ B_1, B_2, ..., B_n$$

  donde n es una constante positiva. La *cabeza* de la regla es A, mientras que el *cuerpo* de la regla viene definida por la conjunción de los objetivos $B_1, B_2, ..., B_n$.

- De acuerdo a la definición previa, puede notarse que un hecho es tan solo un caso particular de una regla, donde n toma el valor cero. Por tanto, se puede entender como una regla con el cuerpo vacío.

- Las variables que aparecen en las reglas son cuantificadas universalmente. De esta forma, el ámbito de las variables abarca toda la regla.

Programas en Prolog: Reglas

- La principal diferencia existente entre las reglas y los hechos es que los hechos son siempre verdaderos mientras que las reglas determinan cosas que podrían ser verdaderas siempre y cuando alguna condición es satisfecha. De esta manera, el cuerpo de la regla es la parte condicional y la cabeza de la regla es la parte de la conclusión.

- La conjunción lógica se define separando los objetivos por medio de comas en el cuerpo de la regla. Ejemplo:

```
feliz(X) :- tienesalud(X) , tienetrabajo(X).
```

- La disyunción lógica se define separando los objetivos por medio de punto y coma en el cuerpo de la regla. Ejemplo:

```
feliz(X) :- rico(X) ; famoso(X).
```

Programas en Prolog: Reglas

A continuación veamos cómo definir relaciones de parentesco entre personas. Comenzamos definiendo un conjunto de hechos simples:

- `madre(andrea, sandra).`
- `madre(sandra, nuria).`
- `padre(antonio, luis).`
- `padre(luis, pepe).`
- `padre(pepe, patricia).`
- `padre(luis, sandra).`

Ahora podemos definir las siguientes reglas:

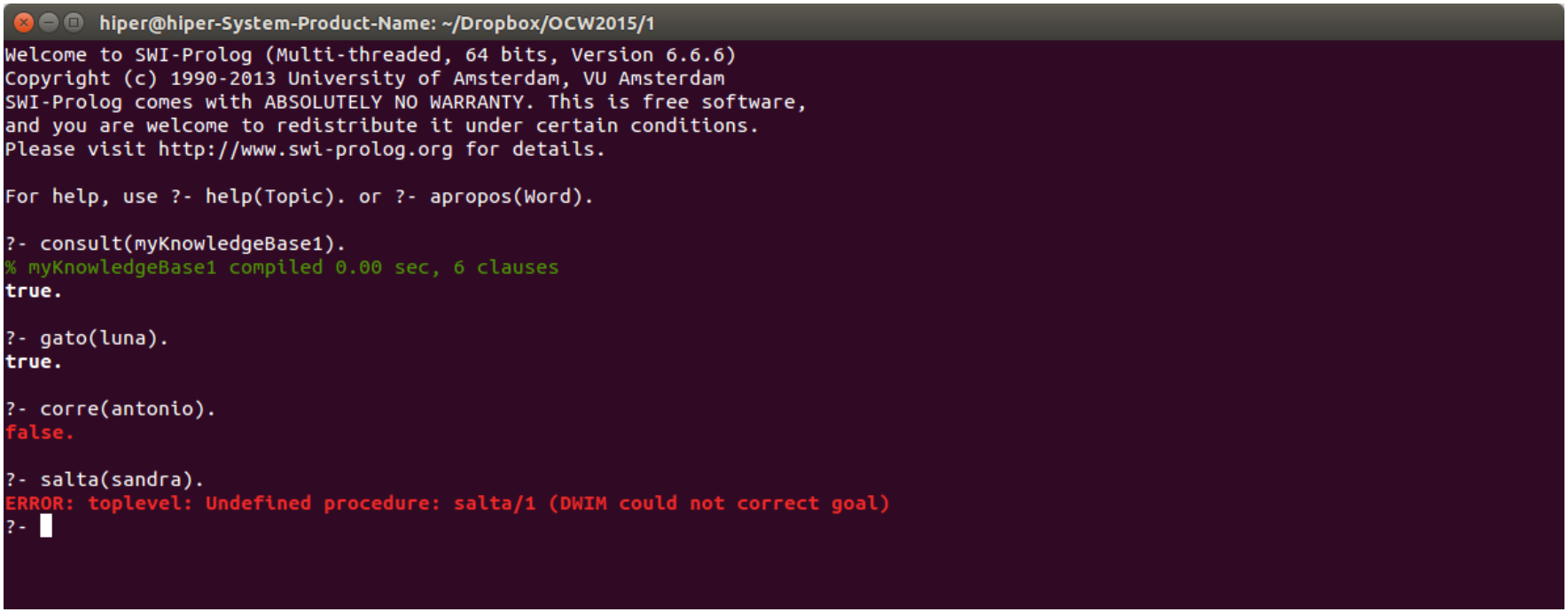
- `abuela(X,Z) :- madre(X,Y), madre(Y,Z).`
- `abuela(X,Z) :- madre(X,Y), padre(Y,Z).`

Programas en Prolog: Ejemplo 1

La base de conocimiento se muestra a continuación:

- `gato(dama).`
- `gato(alaska).`
- `gato(luna).`
- `corre(andrea).`
- `casa.`

Programas en Prolog: Ejemplo 1



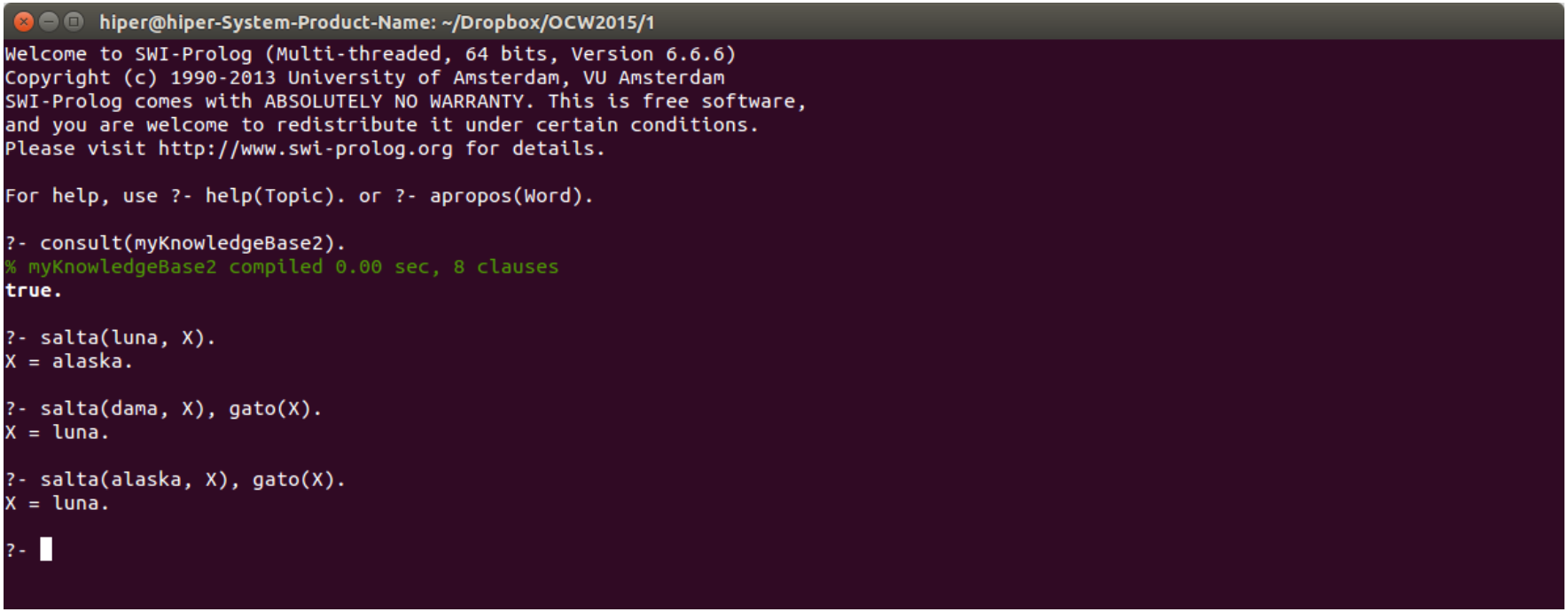
Programas en Prolog: Ejemplo 2

La base de conocimiento se muestra a continuación:

- `gato(dama).`
- `gato(alaska).`
- `gato(luna).`
- `salta(luna, alaska).`
- `salta(dama, luna).`
- `salta(alaska, pelos).`
- `salta(alaska, luna).`

Programas en Prolog: Ejemplo 2

En las consultas realizadas se pueden emplear variables, tal como vemos a continuación:



Recursos: Libros

- Programming in Prolog. Using the ISO Standard
  - Autores: William Clocksin, Christopher S. Mellish
  - Editorial: Springer
- Logic Programming with Prolog
  - Autores: Max Bramer
  - Editorial: Springer
- Language Processing with Perl and Prolog. Theories, Implementation, and Application
  - Autores: Pierre M. Nugues
  - Editorial: Springer
- Prolog Programming for Artificial Intelligence
  - Autores: Ivan Bratko
  - Editorial: Addison Wesley
- The Art of Prolog: Advanced Programming Techniques
  - Autores: Ehud Sterling, Leon Shapiro
  - Editorial: The MIT Press

Recursos: Artículos

- Teaching, learning and using Prolog: Understanding Prolog. Instructional Science. Volumen 19, Issue 4-5, páginas 247-256 (1990)
  - Autores: Paul Brna, Helen Pain y Benedict du Boulay
- Efficient Prolog: a practical tutorial. Artificial Intelligence Review. Volumen 5, Issue 4, páginas 273-286 (1991)
  - Autores: Michael A. Covington
- On the implementation of GNU Prolog. Theory and Practice of Logic Programming. Volumen 12, Special Issue 1-2, páginas 253-282 (2012)
  - Autores: Daniel Diaz, Salvador Abreu y Philippe Codognet

Recursos: Online

- http://www.learnprolognow.org
- http://www.lawebdelprogramador.com/cursos/Prolog/index1.html
- http://www.cpp.edu/~jrfisher/www/prolog_tutorial/contents.html
- http://kti.ms.mff.cuni.cz/~bartak/prolog/index.html
- http://www.cs.bham.ac.uk/~pjh/prolog_course/se207.html
- http://www.amzi.com/ExpertSystemsInProlog

Inteligencia Artificial: Prolog

Introducción al Lenguaje de Programación Prolog

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

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