# Facultad de Ciencias - UNAM
# Lógica Computacional 2023-2
# Práctica 4: Introducción a `Prolog`

Javier Enríquez Mendoza
Ramón Arenas Ayala
Óscar Fernando Millán Pimentel
Kevin Axel Prestegui Ramos

14 de Abril de 2023
**Fecha de entrega:** 20 de Abril de 2023

## 1. Introducción

`Prolog` es un lenguaje de programación lógico que pertenece al estilo declarativo. Entre los principales pilares que sustentan su funcionamiento se encuentra la Lógica de Predicados, estudiada en la sección anterior, así como la unificación de términos y el backtracking.

El enfoque de `Prolog` se trata más de describir hechos conocidos y relaciones entre objetos sobre un problema, y menos de escribir la secuencia de pasos que se necesitan para resolver el problema. De esta manera, por medio de secuencias lógicas `Prolog` logra alcanzar una conclusión lógica partiendo de predicados determinados.

## 2. Objetivo

El objetivo de esta práctica, es poner en práctica los conceptos revisados durante la sesión de laboratorio sobre la programación en el lenguaje Prolog, así como el que los alumnos refuercen el entendimiento de estos y puedan realizar programas complejos utilizando este lenguaje.

## 3. Justificación

El motivo de ser de esta práctica, es el hecho de que como se mencionó anteriormente, `Prolog` será una herramienta de trabajo durante el curso, por lo que es de gran importancia que los alumnos adquieran un nivel de entendimiento sobre los conceptos del lenguaje para el poder desarrollar las prácticas posteriores de manera correcta.

Es importante recalcar, este no es un curso de programación en `Prolog`, ni nos enfocaremos solo en conceptos de programación de `Prolog`, este será nuestra herramienta para implementar conceptos de lógica computacional.

# 4. Desarrollo de la práctica

## 4.1. Instalación de SWI - Prolog

Antes de comenzar a escribir programas en cualquier lenguaje de programación, es necesario instalar, ya sea, un compilador o un intérprete. Para el caso de `Prolog`, existen diversos compiladores, uno de los más conocidos y el cual se usa en esta sección es `SWI-Prolog`.

Para el caso de Windows y MacOS existen archivos binarios que se pueden descargar en `https://www.swi-prolog.org/download/stable`. En el caso de Linux, los detalles de la instalación se encuentran en `https://www.swi-prolog.org/build/unix.html`.

Una vez concluida la instalación, se puede verificar que esta haya sido exitosa ejecutando el comando `swipl -version` , lo que mostrará la versión instalada.

`SWI-Prolog` también funciona como un ambiente interactivo, el cual puede iniciarse ejecutando el comando `swipl` , lo que mostrará un mensaje de bienvenida.

Entre los comandos más útiles para interactuar con el ambiente se tienen:

- `help(X).`; que sirve para mostrar más información acerca de X.
- `halt.`; detiene la ejecución de `SWI-Prolog`.
- `[archivo1, archivo2, ...].`; que sirve para cargar las definiciones de archivos y poder usarlas en el ambiente. Alternativamente, también puede cargar el programa pasando el nombre del archivo como parámetro a `SWI-Prolog: swipl -s program.pl.`
- `make.`; para actualizar las definiciones de algún archivo previamente cargado en el ambiente.

Es importante destacar que el punto al final de cada comando es importante, de esta manera `Prolog` sabe que ahí termina el comando.

## 4.2. Conceptos básicos.

Para poder programar en `Prolog`, es importante comprender varios conceptos básicos, además de entender que aunque pertenece al estilo declarativo, al igual que `Haskell`, este pertenece al **paradigma lógico**, por lo que el estilo de programación será distinto.

Los conceptos básicos para poder programar en `Prolog` son:

- **Términos:** Semánticamente, un término representa a un objeto o a un individuo del universo del discurso. Incluyen términos simples o átomos como números, variables y constantes. Y términos estructurados o estructuras donde se incluyen las funciones y las listas.

- **Predicados:** Un predicado aplicado a un objeto representa una propiedad de ese objeto, y puede evaluarse como «verdadero» o «falso». También existen las literales, las cuales son predicados o predicados negados.
- **Hechos:** Un hecho es un predicado que no contiene ninguna variable entre sus argumentos. Por lo que `Prolog` lo considera como algo que se declara verdadero.
- **Reglas:** Una regla sirve para representar conocimiento que en lenguaje natural se expresa mediante una sentencia condicional. En general, una regla tiene una «cabeza» y un «cuerpo». La cabeza es un predicado, y el cuerpo una conjunción de literales.
- **Consultas:** Una consulta se escribe como un predicado o como una conjunción de predicados. Si la consulta no contiene variables, entonces la máquina responde «YES» o «NO» según que el predicado o la conjunción sean verdaderos o no. Si contiene variables, responde con el valor o los valores para los cuales se hace verdadero.

# 5. Desarrollo de la práctica

## 5.1. Ejercicios básicos

Para practicar los conceptos anteriores y los vistos en clase, realizaremos unos ejercicios básicos:

- Realiza un programa que contenga en la base de conocimiento los signos del Zodiaco. Por ejemplo: `horoscopo (aries, 21, 3, 19, 4)`. A partir de ahí, escribir reglas que permitan calcular el signo del Zodiaco para un día y un mes concreto, por ejemplo: `?- signo(Dia, Mes, Signo)`.
  **Test:**

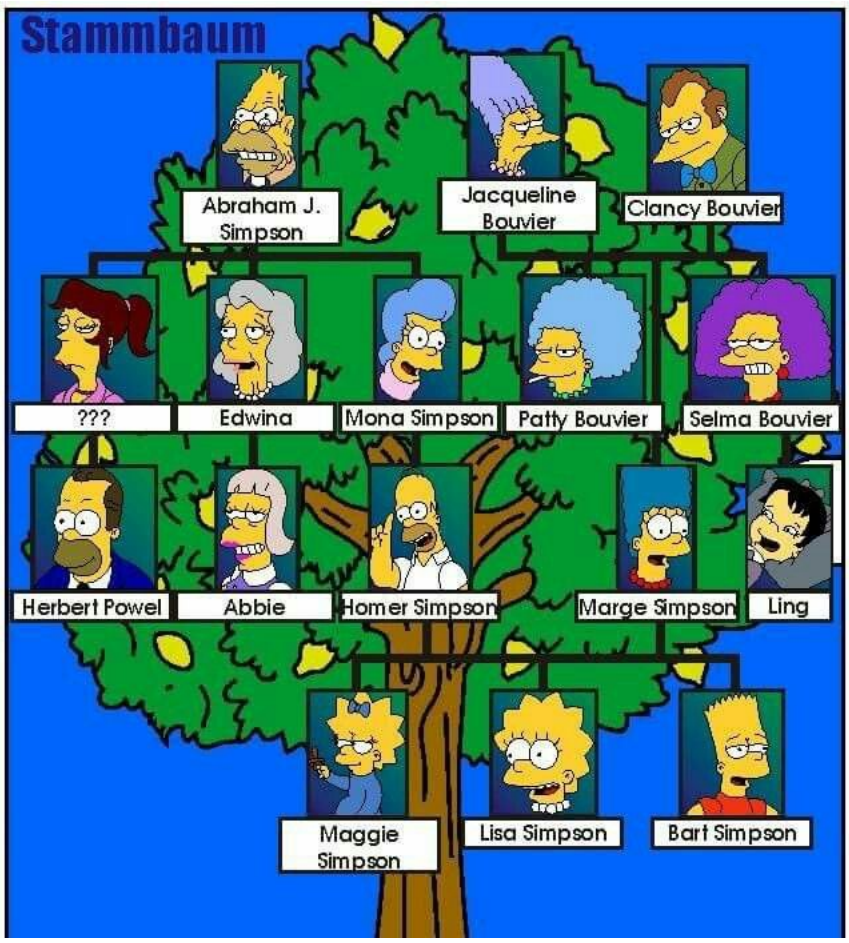
  1. ```
     ?- signo(20,2,X).
     X= Piscis.
     False
     ```
  2. ```
     ?- signo(4,1,X).
     X= Capricornio.
     False
     ```

- Definir procedimiento (predicado) para la siguiente operación con listas, sin utilizar el correspondiente operador `Prolog`: `longitud(L, Num)` Determina el número de elementos de L.
  **Test:**

  1. ```
     ?- longitud([1,2,3,4], X).
     X= 4.
     False
     ```
  2. ```
     ?- longitud([a1,a2,a3,a4,a5], X).
     X= 5.
     False
     ```

## 5.2. Árbol genealógico

Ahora, realizaremos un ejercicio un poco más complejo, creando un conjunto grande de conocimiento y varias reglas para representar un árbol genealógico.




Utilizaremos este árbol genealógico, como base de un programa en `Prolog`. Para esto, se deben definir **única y exclusivamente lo siguiente:**

- **Hechos**
  Género: `hombre("Bart Simpson"), mujer("Lisa Simpson")`
  Progentitor: `progenitor("Homer Simpson", "Bart Simpson"), progenitor("Marge Simpson", "Bart Simpson")`
  Pareja: `pareja("Homer Simpson", "Marge Simpson"`
- **Reglas**: A partir de esto, se deben generar las siguientes reglas: `padre, madre, hermanos, hermano, hermana, esposo, esposa, suegro, suegra, cuñados, cuñado, cuñada, abuelo, abuela, nieto, nieta, tio, tia, primo, prima`.

Realizado lo anterior, se podrá comprobar con las siguientes pruebas.

**Test:**

1. ?- padre(“Bart Simpson“, X).
   X= “Homer Simpson“.
   False
2. ?- madre(“Bart Simpson“, X).
   X= “Marge Simpson“.
   False
3. ?- hermanos(“Maggie Simpson“, X).
   X= “Bart Simpson“.
   X= “Lisa Simpson“.
   False
4. ?- hermano(“Maggie Simpson“, X).
   X= “Bart Simpson“.
   False
5. ?- hermana(“Marge Simpson“, X).
   X= “Selma Bouvier“.
   X= “Patty Bouvier“.
   False
6. ?- esposo(“Jacqueline Bouvier“, X).
   X= “Clancy Bouvier“.
   False
7. ?- esposa(“Homer Simpson“, X).
   X= “Marge Simpson“.
   False
8. ?- suegro(“Homer Simpson“, X).
   X= “Clancy Bouvier“.
   False
9. ?- suegra(“Homer Simpson“, X).
   X= “Jacqueline Bouvier“.
   False
10. ?- cuñados(“Marge Simpson“, X).
    X= “Abbie“.
    X= “Helbert Powel“. False
11. ?- cuñado(“Selma Bouvier“, X).
    X= “Homer Simpson“.
    False
12. ?- cuñada(“Homer Simpson“, X).
    X= “Selma Bouvier“.

```
X= “Patty Bouvier“.
False
```

13. ```
    ?- abuelo(“Bart Simpson“, X).
    X= “Abraham J. Simpson“.
    X= “Clancy Bouvier“.
    False
    ```

14. ```
    ?- abuela(“Bart Simpson“, X).
    X= “Mona Simpson“.
    False
    ```

15. ```
    ?- nieto(“Abraham J. Simpson“, X).
    X= “Bart Simpson“.
    False
    ```

16. ```
    ?- nieta(“Clancy Bouvier“, X).
    X= “Lisa Simpson“.
    X= “Maggie Simpson“.
    X= “Ling“.
    False
    ```

17. ```
    ?- tio(“Bart Simpson“, X).
    X= “Herbet Powel“.
    False
    ```

18. ```
    ?- tia(“Bart Simpson“, X).
    X= “Abbie“.
    X= “Patty Bouvier“.
    X= “Selma Bouvier“.
    False
    ```

19. ```
    ?- primo(“Bart Simpson“, X).
    False
    ```

20. ```
    ?- prima(“Bart Simpson“, X).
    X= “Ling“.
    False
    ```

## 6. Especificaciones de entrega

- **GitHub Classroom:** Para realizar la entrega de la práctica se utilizará la plataforma de GitHub Classroom. `https://classroom.github.com/a/6509fEZB`
- **Equipos:** La práctica puede realizarse en equipos de hasta tres personas. Esto siempre y cuando se respete la política de trabajo colaborativo y podamos ver que ambos miembros del equipo contribuyen en la elaboración de la práctica. En caso contrario, las prácticas serán exclusivamente individuales.

- **Fecha de entrega:** La fecha de entrega será la indicada al principio de este documento. No se recibirá ninguna práctica en fechas posteriores a la fecha indicada, al menos que el profesor indique una prórroga.
- **Flujo de trabajo:** Se recomienda llevar un flujo de trabajo adecuado, en el cual se realicen commits constantemente y no hacer todo al final y en un solo commit. Esto ya que de esta manera podemos comprobar la colaboración de todos los miembros del equipo, dar feedback a lo largo del tiempo de trabajo para mejorar la entrega de las prácticas, facilitar la resolución de dudas, etc.
- **Evaluación:** Para la evaluación la práctica se someterá a un conjunto de pruebas, y además se realizará una revisión del código. Comprobándose que se cumplan las condiciones indicadas para cada ejercicio.
- **Compilado:** Cualquier práctica que al ser descargada no compile, **será evaluada con una calificación de 0**.
- **Datos personales:** Se debe agregar un documento `README.md` en el directorio raíz de sus repositorios, con sus datos personales. En caso de no ser indicados, la calificación no podrá ser asignada.
- **Limpieza y estructura:** en caso de no tener una estructura clara y limpieza dentro de sus repositorios, la calificación se verá afectada negativamente.

## 7. Sugerencias y Notas

- **Dudas**: Pueden preguntar sus dudas a través de telegram, o correo por el canal que ustedes deseen. Pero les recomendamos preguntar a través del grupo de telegram para que sus demás compañeros también puedan aclarar dudas similares a la suya.
- **Limitaciones**: Pueden utilizar funciones predefinidas siempre y cuando su utilización no derive en que se pierda el objetivo académico del ejercicio que se está realizando. Si se tiene duda acerca del poder utilizar algo, pueden preguntarlo.

Buena suerte a todos! ☺☺☺