# Programación para la Inteligencia Artificial
## Presentación del Curso

Dr. Alejandro Guerra-Hernández
*aguerra@uv.mx*

Maestría en Inteligencia Artificial

**IIIA** Instituto de Investigaciones en Inteligencia Artificial
Universidad Veracruzana
*Campus Sur, Calle Paseo Lote II, Sección 2a, No 112*
*Nuevo Xalapa, Xalapa, Ver., México 91097*

21 de agosto de 2023

## 1. Introducción

Este documento presenta la experiencia educativa de Programación para la Inteligencia Artificial (PIA) del Instituto de Investigaciones en Inteligencia Artificial (IIIA) en la Universidad Veracruzana (UV). Nuestro tema de estudio son los paradigmas de programación que la Inteligencia Artificial (IA) ha cultivado como herramientas de desarrollo y sujeto de estudio, *i.e.*, la Programación Lógica y la Funcional. Dada esta dualidad, se abordan tanto los fundamentos teóricos de ambos paradigmas, *e.g.*, el principio de resolución en programas y metas definitivos, y el cálculo lambda; como su aplicación en problemas típicos de la IA, *e.g.*, la resolución de problemas, las búsquedas heurísticas, el aprendizaje automático, la bio-informática, los sistemas de recomendación basados en datos, etc. El resto del documento presenta los objetivos del curso; la manera de evaluar el desempeño en el mismo; los recursos y herramientas que utilizaremos; el calendario de las sesiones; y la bibliografía básica contemplada.

## 2. Objetivos

1. El estudiante identificará los conceptos fundamentales de la Programación Lógica y la Programación Funcional, así como su relación con la IA.

2. El estudiante adquirirá las habilidades para resolver problemas complejos con el lenguaje de Programación Lógica Prolog.

3. El estudiante adquirirá las habilidades para resolver problemas complejos con el lenguaje de Programación Funcional Lisp.

4. El estudiante podrá generalizar estas habilidades usando otros lenguajes de programación que ofrezcan funcionalidades lógicas o funcionales.

## 3. Evaluación

La nota final del curso será calculada de la siguiente forma:

- Las tareas cubren un 70 % de la nota final.
- Un proyecto integrador cubre el 30 % de la nota final.

Para obtener una nota aprobatoria en el curso, el alumno deberá haber aprobado todas estas evaluaciones, *i.e.*, haber obtenido una nota mayor o igual a 70/100 en cada una de las tareas realizadas, el examen y el proyecto integrador.

### 3.1. Tareas

Las tareas pueden requerir investigación bibliográfica, la resolución de ejercicios teóricos, la experimentación en computadora y/o exposiciones frente a grupo. La entrega y evaluación de éstas se realizará conforme a los siguientes lineamentos:

**Entrega.** Las tareas se entregan al inicio de clase del día designado para ello. El mérito de las mismas decrece 25 % por cada 24 horas de retraso. El calendario del curso marca las fechas tentativas para cada tarea, su entrega suele ser dos semanas más tarde.

**Formato.** Las tareas se deben procesar con LaTeX, siguiendo la plantilla de este mismo documento. En todas las partes que involucran código

de computadora o algoritmos, estos deberán documentarse apropiadamente con comentarios y ejemplos de corridas. En algunos casos los ejercicios propuestos indican explícitamente las corridas que deben probarse. Por ejemplo, Si se les pide que implementen en Prolog un predicado `allPerms` que regrese todas las permutaciones de una lista y lo prueben con las lista `[1,2,3]`, la solución debería reportarse como sigue:

```
%%% allPerms computa todas las permutaciones de los elementos
%%% de la lista L en la lista de listas L2.
%%% ?- allPerms([1,2,3],Resp).
%%% Resp = [[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1],
%%%         [3, 1, 2], [3, 2, 1]].

allPerms(L,L2) :-
  findall(Perm, permutation(L,Perm), L2).
```

**Evaluación.** Cada ejercicio de las tareas tiene asignado el puntaje a obtener. Para que un ejercicio o pregunta de la tarea sea evaluado, deberá estar resuelto completamente. Es más redituable invertir el tiempo en contestar una pregunta de manera correcta y completa, que responder a dos de manera incompleta.

**Plagio.** Cualquier forma de plagio causa la expulsión definitiva del curso, y por consiguiente, de la maestría. Esto incluye: Reportar trabajo de otros como propio y no citar pertinentemente las referencias usadas y el código de otros. La responsabilidad en el uso de IAs generativas recae en el alumno, *i.e.,* la IA no plagia, plagia el alumno; la IA no se equivoca, se equivoca el alumno que no la corrigió.

### 3.2. Proyecto integrador

Se trata de un proyecto práctico que requiere la aplicación las diversas técnicas introducidas en el curso, posiblemente complementado con el contenido de otros cursos de este semestre. Se comienza a definir a la mitad del mismo para tratar de considerar los intereses de investigación de los estudiantes. Las fechas de las revisiones parcial y final del proyecto integrador aparecen más adelante en este documento, en el calendario de actividades.

## 4. Recursos del curso

Los recursos del curso incluye algunas páginas web de soporte, los lenguajes de programación a utilizar y diversos ambientes de desarrollo. En lo que sigue, las ligas llevan a estos recursos.

### 4.1. Soporte

- La página web del curso es la referencia de soporte por excelencia.
- Usaremos como complemento el libro The Power of Prolog de Markus Triska, que está disponible como una página web corriendo en un servidor implementado en Prolog.
- The missing semester es un curso sobre diversas herramientas computacionales que todo el mundo espera que sepas usar ¡Y éste no es el caso! Muy útil para todas las experiencias educativas del programa.
- Overleaf debería ser la forma más sencilla de comenzar a trabajar con LaTeX. Su principal limitante es que al ser un editor en línea, requiere una conexión a internet para su uso. Si esto no es posible, instalen LaTeX en el sistema operativo en que estén trabajando.

### 4.2. Lenguajes de programación básicos

- SWI Prolog.
- SBCL Lisp.

### 4.3. Otros lenguajes de programación

- Un dialecto de Lisp, con un ambiente de desarrollo didáctico y buena documentación: Racket Scheme.
- Un Prolog más de frontera, implementado en Rust: Scryer Prolog.

### 4.4. Ambientes de desarrollo

- Emacs.
- Visual Studio Code.

## 5. Calendario

El curso esta organizado en un semestre con dos sesiones presenciales los martes y jueves de 10:00 a 12:00 hrs. En esta ocasión, las sesiones presenciales se llevarán a cabo en las instalaciones del IIIA, Edificio C, aula C8 (penúltimo piso al fondo). Los temas a cubrir en el curso se organizan como se muestra en el Cuadro 1. T1 indica la fecha tentativa en que será encargada la tarea 1, etc. P1 indica la primer revisión de definición del proyecto integrador. P2 indica la revisión final del mismo.

| Fecha | Sesión | Tarea |
|---|---|---|
| 22/08/2023 | Presentación del curso | |
| 24/08/2023 | Paradigmas de programación | |
| 29/08/2023 | | |
| 31/08/2023 | Programación lógica | |
| 30/09/2023 | | |
| 05/09/2023 | | |
| 07/09/2023 | | T1 |
| 12/09/2023 | Prolog | |
| 14/09/2023 | | |
| 19/09/2023 | | |
| 21/09/2023 | | T2 |
| 26/09/2023 | Prolog en IA | |
| 28/09/2023 | | |
| 03/10/2023 | | |
| 05/10/2023 | | |
| 10/10/2023 | | T3 |
| 12/10/2023 | Programación funcional | |
| 17/10/2023 | | |
| 19/10/2023 | | |
| 24/10/2023 | Lisp | |
| 26/10/2023 | | |
| 31/10/2023 | | P1 |
| 07/11/2023 | | T4 |
| 09/11/2023 | | |
| 14/11/2023 | Lisp en IA | |
| 16/11/2023 | | |
| 21/11/2023 | | |
| 23/11/2023 | | T5 |
| 28/11/2023 | | |
| 30/12/2023 | | |
| 04/01/2024 | | P2 |

Cuadro 1: Sesiones del semestre

## 6. Bibliografía

Bratko (2012) es la referencia básica para Prolog y sus aplicaciones a la IA. Norvig (2001) hace lo propio con Lisp. Seibel (2005), Weitz (2016) y Watson (2020) proveen una revisión más moderna de Lisp, como el libro de Triska y el de Tate (2022) lo hacen con Prolog. Los fundamentos teóricos de la programación lógica se revisan con el apoyo de Nilsson and Maluszynski (2000); Julián Iranzo and Alpuente Frasnedo (2007); Genesereth and Chaudhri (2020) y los de la programación funcional con los libros de Kluge (2005) y Michaelson (2011). Stelly (2021) provee una divertida introducción a Scheme, un dialecto de Lisp, y su ambiente de desarrollo Racket. Warren et al. (2023) presentan el estado del arte de la programación lógica en su 50o aniversario y las perspectivas futuras de este paradigma de programación. Algunos aspectos de la programación funcional han sido adoptados por lenguajes de programación que no son estrictamente funcionales, Matuszek (2023) hace una revisión esto para Python, Java y Scala.

## Referencias


Bratko, I. (2012). *Prolog programming for Artificial Intelligence*. Pearson, fourth edition.

Genesereth, M. and Chaudhri, V. K. (2020). *Introduction to Logic Programming*, volume 44 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers, San Rafael, CA, USA.

Julián Iranzo, P. and Alpuente Frasnedo, M. (2007). *Programación Lógica: Teoría y Práctica*. Pearson Educación S.A., Madrid, España.

Kluge, W. (2005). *Abstract Computing Machines: A Lambda Calculus Perspective*. Springer-Verlag, Berlin Heidelberg New York.

Matuszek, D. (2023). *Quick Functional Programming*. Quick Programming Series. CRC Press, Boca Raton, FL, USA.

Michaelson, G. (2011). *An Introduction to Functional Programming through Lambda Calculus*. Dover Publications, Inc., Mineola, New York, USA, Dover edition.

Nilsson, U. and Maluszynski, J. (2000). *Logic, Programming and Prolog*. John Wiley & Sons Ltd, 2nd edition.

Norvig, P. (2001). *Paradigms of Artificial Intelligence Programming: Case Studies in Common Lisp*. Morgan Kauffman Publishers, Los Altos, CA, USA, 6th edition.

Seibel, P. (2005). *Practical Common Lisp*. Apress, USA.

Stelly, J. W. (2021). *Racket Programming the Fun Way: From Strings to Turing Machine*. No Starch Press, San Francisco, CA, USA.

Tate, B. A. (2022). *Programmer Passport: Prolog*. The Pragmatic Programmers, LLC, Raleigh, NC, USA.

Warren, D. S., Dahl, V., Eiter, T., Hermenegildo, M. V., Kowalski, R., and Rossi, F. (2023). *Prolog: The Next 50 years*. Number 13900 in Lecture Notes in Artificial Intelligence. Springer Nature Switzerland AG, Cham, Switzerland.

Watson, M. (2020). *Loving Common Lisp or the Savvy Programmer's Secret Weapon*. Leanpub, USA.

Weitz, E. (2016). *Common Lisp Recipes: A Problem-Solution Approach*. Apress, New York, NY, USA.