

Metodologías de Programación II (Funcional): Sobre este curso.

Dr. Alejandro Guerra-Hernández
Departamento de Inteligencia Artificial
Universidad Veracruzana
Facultad de Física e Inteligencia Artificial
Sebastián Camacho No 5, Xalapa, Ver., México 91000
aguerra@uv.mx
www.uv.mx/aguerra

9 de enero de 2012

Bienvenidos al curso de Metodología de Programación II, en la Maestría en Inteligencia Artificial de la Universidad Veracruzana. El tema central de este curso es la **programación funcional**, y complementa al curso de Metodología de Programación I, cuyo tema es la programación lógica. El material está organizado para ofrecer un curso de 45 horas, durante un cuatrimestre, con dos sesiones de dos horas por semana.

1. Objetivos

El **objetivo** de este texto es introducir al estudiante a las técnicas y métodos de la programación funcional en el contexto de la Inteligencia Artificial. Para ello, la primera parte del mismo se ha organizado como un taller práctico de programación en Lisp; mientras que la segunda parte hace énfasis en el aspecto teórico de este paradigma de programación, ilustrando con ejemplos en el dialecto de ML conocido como OCaml.

2. Evaluación

El estudiante será **evaluado** conforme a su participación en clase, la calidad de sus tareas y el resultado que obtenga en la elaboración de un proyecto final, que

involucra las competencias que un estudiante debe adquirir en este curso. La nota final, será calculada, aproximadamente, de la siguiente forma:

- 4 tareas que cubren el 60% de la nota final. Las tareas incluyen escritura de ensayos, exposiciones y, por supuesto, ejercicios de programación.
- El proyecto final cubre el 40% de la nota final.

Para obtener una **nota aprobatoria**, el alumno deberá haber obtenido notas aprobatorias en cada uno de las tareas a realizar y el proyecto final, es decir, no aprobar cualquier elemento parcial de la evaluación, implica no aprobar el curso.

3. Tareas

Durante el curso, se asignarán cuatro tareas. Las tareas deberán entregarse a las 11:00 am del día designado (generalmente al encargarse la tarea siguiente, por ejemplo, la tarea T1 se entregará el día que será encargada la tarea T2, y así sucesivamente). El mérito del trabajo decrece 25% por cada 24 horas de retraso. Las tareas pueden requerir investigación bibliográfica y experimentación en la computadora (Ocaml, Lisp). Todos los trabajos deberán incluir al pie de cada página: i) el nombre del estudiante; ii) el nombre del curso; iii) el número de la tarea; y iv) el número de la página. Sólo se acepta papel carta 8'x11'. En todas las partes que involucran código, éste deberá ser documentado apropiadamente. Los trabajos son evaluados de la siguiente manera: Para que un ejercicio o pregunta reciba créditos, más del 50% deberá estar resuelta de manera correcta o completa (es más reduible invertir el tiempo en contestar una pregunta de manera correcta y completa, que responder a dos de manera parcial).

La discusión entre los estudiantes es uno de los aspectos más enriquecedores en un programa de maestría. Es sumamente importante que discutan entre ustedes las tareas, pero toda discusión al respecto deberá ser de manera oral. En ningún momento podrá un estudiante consultar las notas de trabajo escrito de otro. Cualquier anomalía en este sentido, causa **expulsión definitiva** del curso. La fecha de entrega de las tareas aparece en la sección de calendario de este prefacio.

4. Proyecto final

El proyecto final gira en torno al algoritmo de inducción de árboles de decisión ID3. En la primera parte de curso, implementaremos una primer versión de este algoritmo. El proyecto final consiste en implementar una serie de mejoras al mismo, que pueden estar relacionadas con:

- Preprocesamiento de datos (Discretización, formatos de datos, etc.);
- Eficiencia de la implementación (Concurrencia, optimización, representación de datos, poda, etc.);
- Usabilidad (Interfases gráficas, comandos nuevos, salida, etc.);
- Aplicación, encontrar un problema relevante en nuestro contexto donde aplicar el sistema implementado.

Para ello, la primer parte del curso está orientada a que el estudiante adquiera las habilidades necesarias para contender con este proyecto.

5. Material del curso

El estudiante es libre de elegir la implementación de Lisp que crea conveniente, aunque se recomienda usar **LispWorks 6.0**¹ para el proyecto final del curso (contamos con una licencia comercial en OS X). Otras implementaciones disponibles incluyen: **Clozure**², **Allegro**³ y **SBCL**⁴, entre otras. Con respecto a ML utilizaremos el dialecto **OCaml 3.12.0**⁵.

Las notas del curso están en: <http://www.uv.mx/aguerra/mpii.html>, separadas por sesiones, con el material complementario correspondiente.

Se les solicita usar **Latex**⁶ para generar los reportes con los resultados de sus tareas. Tanto Latex, como Lisp y Ocaml pueden integrarse al editor **Emacs**⁷. El archivo de configuración `.emacs` debería contener algunas líneas como las siguientes:

```

1  ;;; slime (setting Lisp for aquaemacs)
2
3  (add-hook 'slime-mode-hook '
4           imenu-add-menubar-index)
5
6  (setq inferior-lisp-program "~/Documents/lisp/
7           lispworks ")

```

¹<http://www.lispworks.com>

²<http://trac.clozure.com/ccl>

³<http://www.franz.com/>

⁴<http://www.sbcl.org/>

⁵<http://caml.inria.fr/>

⁶<http://www.latex-project.com>

⁷<http://www.gnu.org>

```

6      lisp-indent-function '
          common-lisp-indent-function
7      common-lisp-hyperspec-root
8      "file:///Applications/LispWorks%205.1/
          Library/lib/5-1-0-0/manual/online/web/
          CLHS/"
9      slime-start-up-animation t)
10
11     (setq auto-mode-alist
12           (cons '("\\.lsp" . lisp-mode)
                  auto-mode-alist))
13
14     ;;; configuration for Ocaml
15
16     (add-to-list 'load-path "/Users/aguerra/Documents/
          elisp/tuareg-mode-1.45.6/")
17
18     (setq auto-mode-alist (cons '("\\.ml\\w?" .
          tuareg-mode) auto-mode-alist))
19     (autoload 'tuareg-mode "tuareg" "Major mode for
          editing Caml code" t)
20     (autoload 'camldebug "camldebug" "Run the Caml
          debugger" t)
21     (if (and (boundp 'window-system) window-system)
22         (require 'font-lock))

```

La línea 3 añade una entrada al menú de Emacs que lista las funciones definidas por el usuario. La línea 5 especifica la implementación de Lisp que se va a usar. La línea 8 define el camino a la documentación de Lisp en formato HTML. Slime y Tuareg-Mode deben cargarse de internet por separado.

6. Calendario

Este año las sesiones se llevarán a cabo los lunes y los miércoles de 11:00 a 13:00 hrs. El contenido y **calendario** de las sesiones es el siguiente:

Fecha	Tema	Tarea
09/01/2012	Introducción a la Programación Funcional	
11/01/2012	Introducción a Lisp 1	
16/01/2012	Introducción a Lisp 2	
18/01/2012	Recursividad y Listas en Lisp	T1
23/01/2012	Macros 1	
25/01/2012	Macros 2	
30/01/2012	Aspectos no funcionales de Lisp	T2
01/02/2012	Práctica 1: Un filtro de Spam	
06/02/2012	Práctica 2: Programación Web	
08/02/2012	Práctica 3: Árboles de decisión	T3
13/02/2012	Introducción a Ocaml	
15/02/2012	Listas y tipos de datos definidos por el usuario	
20/02/2012	Computaciones repetitivas	
22/02/2012	Tipos de datos concretos y recursivos	T4
27/02/2012	Cálculo lambda 1	
29/02/2012	Cálculo lambda 2	
26/03/2012	Entrega de Proyectos	

Referencias básicas

1. E. Chailloux, P. Manoury, and B. Pagano. *Developing Applications With Objective Caml*. O'Reilly, Paris, France, 2000.
2. P. Graham. *On Lisp: Advanced Techniques for Common Lisp*. Prentice Hall International, 1993.
3. P. Graham. *ANSI Common Lisp*. Prentice Hall Series in Artificial Intelligence. Prentice Hall International, 1996.
4. A. Guerra-Hernández. *Metodologías de Programación II: Programación Funcional*. <http://www.uv.mx/mpii.html> Edición 2012.
5. W. Kluge. *Abstract Computing Machines: A Lambda Calculus Perspective*. Springer-Verlag, Berlin Heidelberg New York, 2005.
6. P. Norvig. *Paradigs of Artificial Intelligence Programming: Case Studies in Common Lisp*. Morgan Kauffman Publishers, 1992.
7. C. Queinnec. *Lisp in Small Pieces*. Cambridge University Press, Cambridge, UK, 1996.

8. P. Seibel. *Practical Common Lisp*. Apress, USA, 2005.

Xalapa, Ver., México
Enero 2012

*Alejandro
Guerra-Hernández*