2018-19

INTRODUCCIÓN A LA PROGRAMACIÓN CON LISP

PARTICULARIDADES DE LOS PROBLEMAS EN INTELIGENCIA ARTIFICIAL

-Problemas pobremente definidos y especificados inicialmente, aspecto sólo mejorable mediante refinamiento progresivo incremental

-En dominios complejos

-Problemas dependientes del contexto, con estrategias y heurísticas a desarrollar mediante paulatina prueba-error

-Con enfoques simbólicos o sub-simbólicos, más que numéricos

EJEMPLOS

Procesamiento de lenguaje natural, traducción automática, visión por ordenador, control de procesos mal modelizables (como fabricación de cemento), procesos adaptativos o evolucionarios, diagnosis médica, toma de decisiones económicas etc.

CONSECUENCIAS PARA LA PROGRAMACIÓN

-Diferencias con la Ingeniería del software habitual: no se comienza con una especificación precisa

-El trabajo de implementación forma parte del proceso de especificación (progresivo)

-Deseable librar al programador de ocuparse de limitaciones técnicas de construcción (bajo nivel en tipos de datos nuevos, gestión de depósito en memoria etc.)

-Conveniencia de usar un estilo declarativo usando estructuras de datos (como árboles o listas) y operaciones (como ajuste de patrones) *built-in* de alto nivel

-Programación que soporte computación simbólica de alto nivel de abstracción

-Dificultad de compilar eficientemente en máquinas estándar la programación con estilo imperativo, inicialmente, si bien, una vez comprendidos y especificados algunos problemas se pueden hacer, acaso parcialmente, reimplementaciones en imperativo.

PARADIGMAS ALTERNATIVOS SURGIDOS DE LA IA

-***Programación Funcional***, basada en la Teoría de Funciones Recursivas y el λ-cálculo: Lisp (McCarthy, finales de los cincuenta)

-***Programación Lógica***, basada la Lógica Formal: Prolog (Colmerauer, Kowalski, Roussel, años setenta) Nota: Ésta será tratada en la asignatura Conocimiento y razonam. automatizado

PROGRAMACIÓN FUNCIONAL

--Usa ***funciones matemáticas*** (no cambios de estado mediante mutación de variables).

--Evita los ***efectos colaterales*** que pueden tener las "funciones" de la programación imperativa, en la que puede variar el valor de cálculos previos.

--Evita el concepto de ***estado del cómputo***.

--Los programas están constituidos sólo por definiciones de funciones (entendidas, ya se ha dicho, como funciones matemáticas, no funciones concebidas como subprogramas) -La programación consiste en construir definiciones de funciones, acaso combinando otras previas, para cada problema específico

--La computación consiste en evaluar llamadas de funciones e imprimir valores resultantes, como una calculadora manual, pero a un nivel mucho más flexible y potente

--Hay ***transparencia referencial*** (el significado de una expresión depende sólo del significado de sus subespresiones)

--No hay asignación de variables, secuencias ni iteraciones. Todas las repeticiones se efectúan mediante ***funciones recursivas***.

-- Control de flujo: mediante recursión y condicionales, sin secuenciación ni iteración.

--Se basan en el ***λ-Cálculo***, uno de los modelos del concepto de ***Computabilidad***, el más próximo al punto de vista de lo que hoy llamaríamos *software.* Los otros paradigmas de programación están más dirigidos a. la perspectiva de la máquina (deTuring).

--Alto nivel de abstracción: conseguido mediante el uso de la abstracción funcional y la aplicación funcional definidas en el λ-cálculo

--BeneficioS: Alto nivel de modularidad, facilidad para descomponer problemas en subproblemas y para ensamblar los resultados parciales correspondientes

PROGRAMACIÓN FUNCIONAL EN LISP

-***Lisp*** (de LISt- Processing, McCarthy 1958): primer lenguaje funcional

-Diseñado para permitir computación simbólica, usando listas enlazadas como principal estructura de datos

-Principales dialectos en uso: ***Scheme*** y ***Common Lisp***

En este curso, nos ocuparemos preferentemente de Scheme-Lisp, versión ***Racket***

SITIOS WEB

-Sitios web sobre Scheme:

http://www.racket-lang.org/ http://www.gnu.org/software/mit-scheme

-Sitios web sobre Common Lisp:

www.lispworks.com

BIBLIOGRAFÍA SOBRE LISP-SCHEME

- **Van Horn, Barski, Felleisen, *Realm of Racket***, No Starch Press 2013, (disponible en la web)
- **Abelson, Sussman, *Structure and Interpretación of Computer Programs*,** The MIT Press 1996.
- **E. Navas, *Programando con Racket 5*, *Ed. D.E.I. Univ. José Simeón Cañas. 2010***
- **R. Kent Dybvig**, ***The Scheme Programming Language***, The MIT Press (2009) Puede descargarse de: http://www.scheme.com/tspl4/
- **M. Felleisen, R.B. Findler, M. Flatt y S. Krishnamurthi**, ***How to Desing Programs***, ed.The MIT Press (2001). Puede descargarse de : http://htdp.org/
- **D.P. Friedman y M. Felleisen, *The Little Schemer***, The Mit Press, en: http://scottn.us/downloads/The_Little_Schemer.pdf

BIBLIOGRAFÍA SOBRE COMMON LISP

- **Winston-Horn , *LISP,*** Addison-Wesley Iberoamericana (1991)
- **Graham, *ANSI Common Lisp,*** Prentice Hall (1995)

REFERENCIAS SOBRE PROGRAMACIÓN FUNCIONAL

An introduction to the basic principles of Functional Programming
https://medium.freecodecamp.org/an-introduction-to-the-basic-principles-of-functional-programming-a2c2a15c84

Functional Programming:

https://www.revolvy.com/page/List-of-functional-programming-topics

https://www.revolvy.com/folder/Functional-programming/250966