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 evolucionario, 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

-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 Razonamiento Automatizado

PROGRAMACIÓN FUNCIONAL (frente a imperativa):

-Usa funciones con nombre y las anónimas del λ-cálculo ( Church 1936)

-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

-El orden en que se ejecute la evaluación no afecta a su resultado, un valor. No puede haber efectos colaterales debidos a la posición de variables en memoria, al no usarse variables

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

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

-Beneficio: 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

**SITIOS WEB**

-Sitios web sobre Common Lisp:

[www.lispworks.com](http://www.lispworks.com)

-Sitios web sobre Scheme:

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

**BIBLIOGRAFÍA SOBRE LISP-SCHEME**

* **R. Kent Dybvig**, ***The Scheme Programming Language***, ed. 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)