

Universidade de Aveiro

*Introdução à Inteligência Artificial*  
(MIECT)

# Noções de Programação Declarativa

Ano lectivo 2020/2021

Regente: Luís Seabra Lopes

# Programação Declarativa

- Os principais paradigmas de programação declarativa são:
  - Programação funcional
    - baseado no cálculo-lambda
    - a entidade central é a função
  - Programação em lógica
    - baseado na lógica de primeira ordem
    - a entidade central é o predicado

# Paradigma imperativo

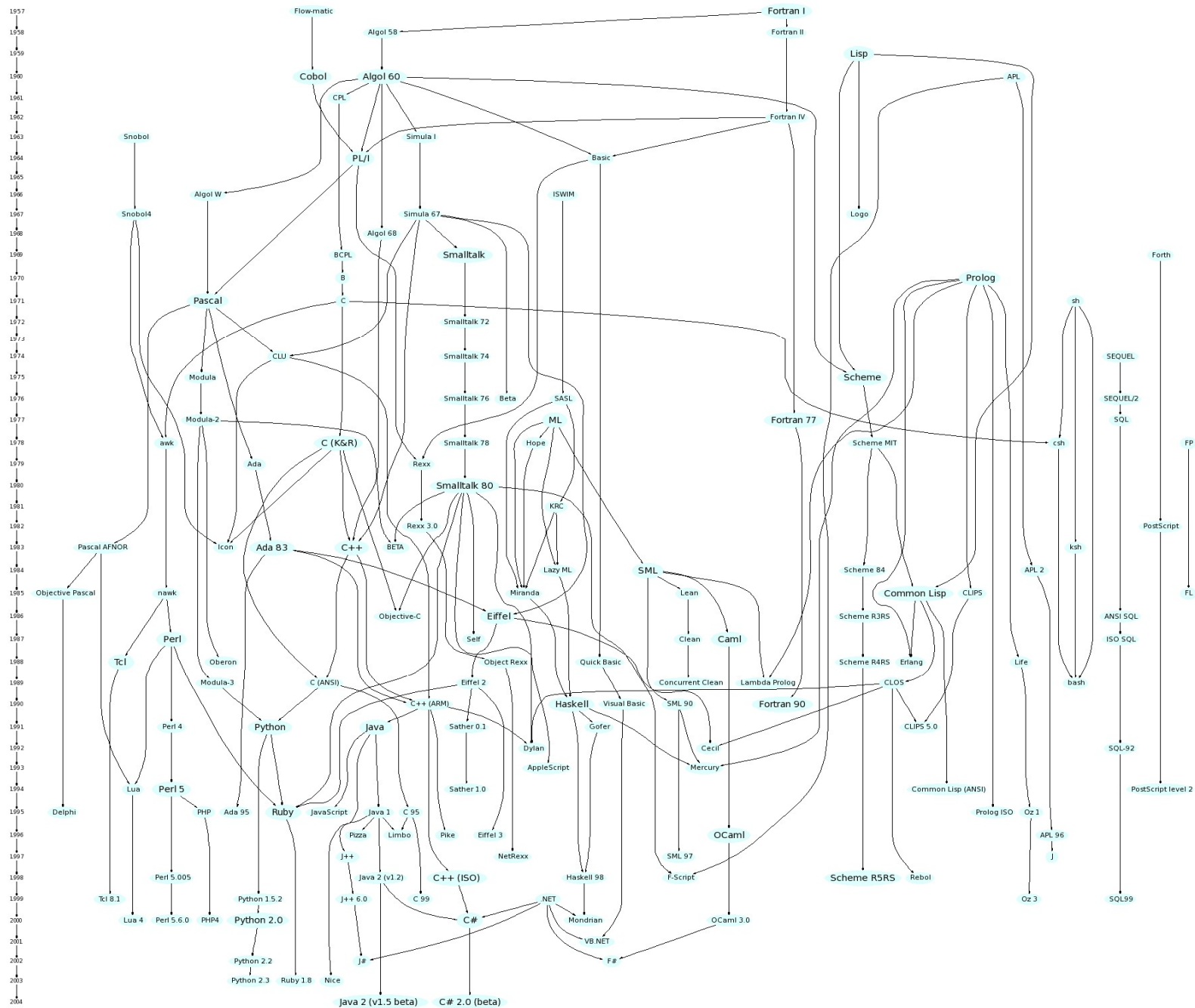
- O fluxo de operações é explicitamente sequenciado
  - Noções de “instrução” e “sequência de instruções”
- Memória
  - Há alterações ao conteúdo da memória (instruções de afectação/atribuição)
  - Pode haver variáveis globais
- Análise de casos: if-then-else, switch/case ...
- Processamento iterativo: while, repeat, for, ...
- Sub-programas: procedimentos, funções

# Paradigma declarativo

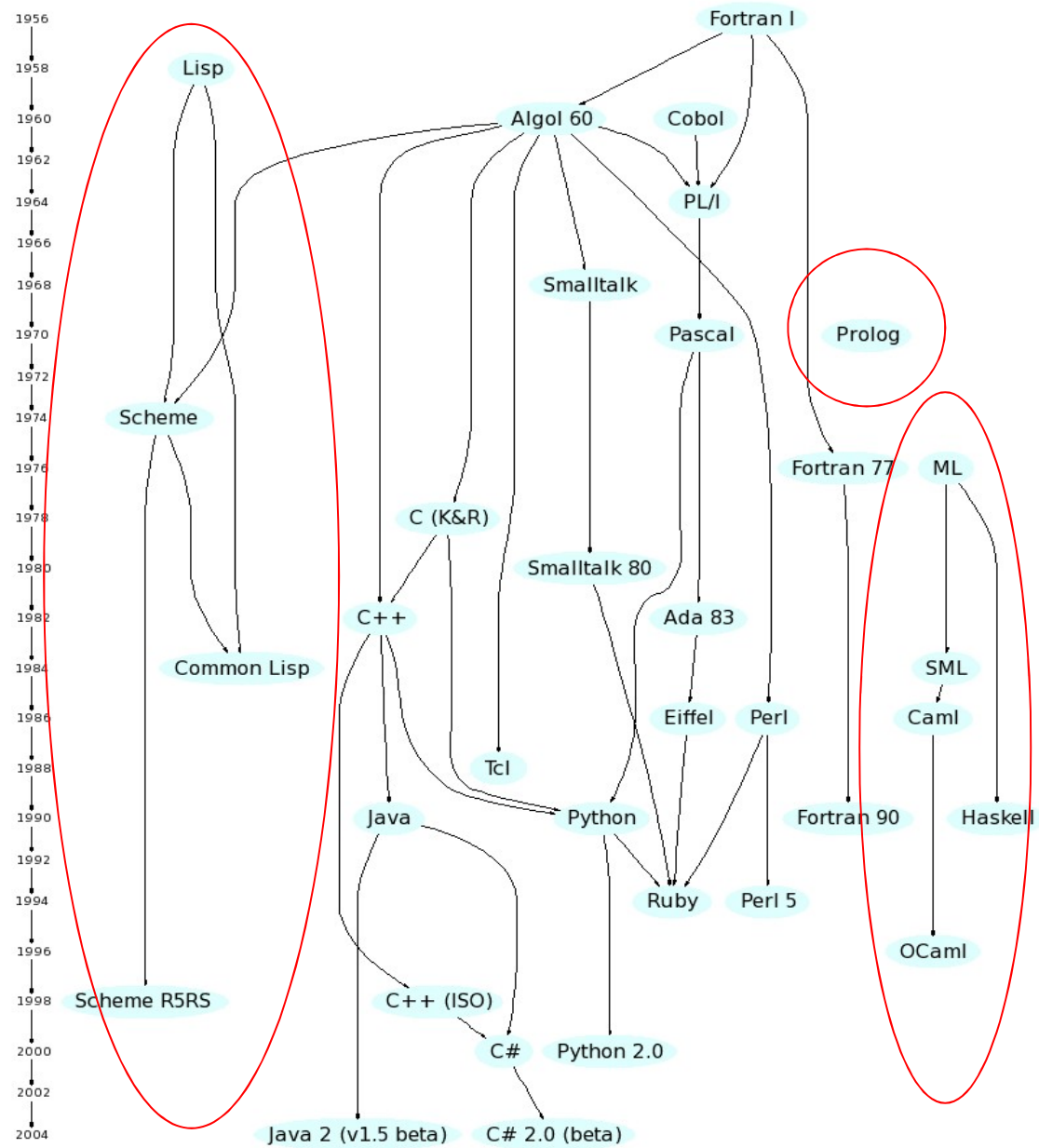
	Funcional	Lógico
Fundamentos	Lambda calculus	Lógica de primeira ordem
Conceito central	Função	Predicado
Mecanismos	Aplicação de funções Unificação uni-direccional Estruturas decisórias	Inferência lógica (resolução SLD) Unificação bi-direccional
Programa	Um conjunto de declarações de funções e estruturas de dados	Um conjunto de fórmulas lógicas (factos e regras)

# Programação Declarativa - História

- 1957 - FORTRAN - A primeira linguagem de programação (desenhada na IBM por John Backus)
  - É uma linguagem imperativa
  - Este paradigma foi sendo aperfeiçoado, dando sucessivamente origem a linguagens como Algol (1960), Pascal (1971) e C (1972).
- 1960 - LISP - Inventada por John McCarty, como linguagem para processamento simbólico em problemas de inteligência artificial
- 1970 – Prolog – Inventada por Colmerauer na Univ. Marselha
- 1978 - ML - Desenhada por Robin Milner, como linguagem de comandos para um sistema de prova da correcção de programas
- 1995 – Mercury –Linguagem que integra os paradigmas lógico e funcional, desenvolvida na Univ. Melbourne



## IIA: Noções de Programação Declarativa



*IIA: Noções de Programação Declarativa*

# Cheiro: Programação Funcional

- Possibilidade de definir funções localmente e sem nome
- Em Lisp:
  - `((lambda (x) (+ (* 2 x) 1)) 6)`
  - Resultado: 13
- Em Caml:
  - `(fun x -> 2*x+1) 6`
  - Equivalente à anterior



# Cheiro: Programação em Lógica

- Um programa é uma teoria sobre um domínio
- Exemplo:
  - homem(socrates).
  - mortal(X) :- homem(X).
- Pergunta:
  - ?- mortal(socrates).
  - Yes

# Recursividade “omnipresente”

Cam1

```
let rec fact(n) =  
  if (n=0)  
  then 1  
  else n*fact(n-1);;
```

Prolog

```
fact(0,1).  
fact(N,FactN):-  
  M is N-1,  
  fact(M,FactM),  
  FactN is N*FactM.
```

```
int fact(int n)  
{  
  if (n==0) return 1;  
  else return n*fact(n-1);  
}
```

C

```
int fact(int n)  
{  
  int f=1, i;  
  for(i=1; i<=n ; i++)  
    f=f*i;  
  return f;  
}
```

C (iterativo)

# Atitude do programador

- A programação declarativa, dada a sua elevada expressividade, é pouco compatível com aproximações empíricas (ou “tentativa-e-erro”) à programação.
- Convem pensar bem na estrutura do programa antes de começar a digitar
- Aconselham-se os seguintes passos:
  - Perceber o problema
  - Desenhar o programa
  - Escrever o programa
  - Rever e testar

# Programação funcional - Características

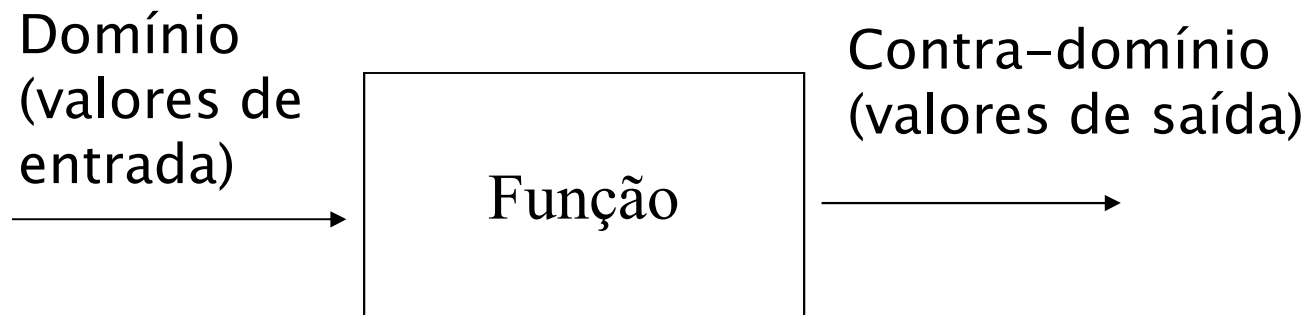
- A entidade central é a função
- A noção de função é directamente herdada da matemática (*ao contrário, nas linguagens imperativas, o que se chama função é por vezes algo muito diferente de uma função matemática*)
- A estrutura de controlo fundamental é a “aplicação de funções”
- A noção de “tipo da função” captura a noção matemática de domínio (de entrada e de saída)
- Os elementos dos domínios de entrada e saída podem por sua vez ser funções

# Origens da Programação Funcional

- Origem na disciplina da lógica matemática, relacionada com o estudo da computabilidade de funções
- 1930 - Alonzo Church inventa o  $\lambda$ -calculus (Cálculo Lambda), um sistema formal para descrever funções computáveis
- Já na era do computador, o conceito de função computável tomou o sentido de “função que pode ser implementada através de um programa de computador”.
- O Cálculo-Lambda permite definir a semântica das linguagens de programação

# Função

- Tem valores de entrada (domínio) e valores de saída (contra-domínio)



# Lambda Calculus

- Sistema formal
  - Alonzo Church e Stephen Cole Kleene em ~1930
- Definir formalmente
  - Funções, aplicação de funções, recursividade
- A mais pequena linguagem universal
  - Tudo o que pode ser programado tem equivalente em Lambda calculus
  - Equivalente à máquina de Turing
- Permite provar matematicamente correcção de programas

# LISP - I

- LISP = LISt Processing
- Das linguagens de programação que tiveram grande divulgação, LISP é a segunda mais antiga
- Listas são usadas para representar quer os dados quer os programas
- A ideia central é a de “Aplicação de Funções”
- Uso intensivo de funções recursivas
- Permite a definição de funções de ordem superior
- Tem estruturas de decisão condicional
- Não tem um sistema de tipos



# LISP - II

## Processamento de listas

- A lista é representada como uma sequência de elementos separados por espaços e delimitada por parentesis
  - Exemplo: (a b c d)
- Operações (entre muitas outras):
  - Adicionar elemento à frente: (cons x L)
  - Obter primeiro elemento: (car L)
  - Obter resto: (cdr L)
  - Portanto: (cons (car L) (cdr L)) dá a lista L

# LISP - III

## Exemplos

- Estrutura de dados
  - (DET 1974 (45 25) (MIECT LTSI MIEET) (MSI))
    - “O DET foi fundado em 1974, tem 45 docentes doutorados, etc..”
- Expressão
  - (f (+ x y) (\* u z))
    - Equivalente a  $f(x+y, u*z)$
- Função anónima
  - (lambda(x y) (sqrt (+ (\* x x) (\* y y))))
- Função com nome:
  - (defun modulo(x y) (sqrt (+ (\* x x) (\* y y))))

# LISP - IV

- LISP foi durante décadas a linguagem dominante no campo da inteligência artificial, e continua a ser usada
- Tradicionalmente interpretada (mas há compiladores há muito tempo também)
- Uma vez que é muito antiga, a linguagem LISP está cheia de “remendos” e tem muitos dialectos:
  - **COMMON LISP** - reúne as características comuns aos principais dialectos do LISP
  - **Scheme** - um dialecto do LISP desenhado de raiz por forma a evitar os tais remendos

# ML

- ML (= MetaLanguage) - começou por ser uma linguagem de interface para um sistema de prova da correcção de programas
- É essencialmente o formalismo do cálculo-lambda com uma sintaxe mais agradável
- Argumentos avaliados antes da respectiva passagem para o interior da função (*call-by-value*)
- Principais dialectos:
  - SML (= Standard ML) – 1984 – Bell Labs, em cooperação com Edimburgo, Cambridge e INRIA, sob a direcção de Robin Milner
  - Caml – 1987 - desenvolvida no INRIA (França)

# Miranda, Haskell

- Constituem um grupo à parte dentro das linguagens funcionais
- Os argumentos são passados não avaliados para o interior das funções – só são avaliados se forem necessários (*lazy evaluation*)
- Principais linguagens:
  - Miranda (1985)
  - Haskell (1990)

# Programação em Lógica

- Um programa numa linguagem baseada em lógica representa uma teoria sobre um problema
- Um programa é uma sequência de frases ou fórmulas representando
  - *factos* - informação sobre objectos concretos do problema / domínio de aplicação
  - *regras* - leis gerais sobre esse problema / domínio
- Implicitamente
  - as frases estão reunidas numa grande conjunção, e
  - cada frase está quantificada universalmente.
- Portanto, ***programação declarativa***.

# A linguagem Prolog

- ‘Prolog’ é acrónimo de ‘Programação em Lógica’
- Desenvolvida circa 1970 em Marselha (Colmerauer) e Edimburgo (Kowalski, Pereira, Warren)
- Execução de um programa Prolog é dirigida pela informação necessária para resolver um problema e não pela ordem das instruções de um programa
  - Um programa Prolog começa com uma pergunta (query)
- Mecanismos centrais:
  - unificação,
  - estruturas de dados baseadas em listas e árvores,
  - procura automática de alternativas

# Prolog - programas

- Factos são fórmulas atómicas, ou seja, fórmulas que consistem de um único predicado. Exemplos:
  - lecciona(lsl, iia).
  - mulher(joana).
  - aluno(alfredo,ect,ua).
- As regras são implicações com um único conseqüente e um ou mais antecedentes. Exemplo:
  - professor(X) :- lecciona(X,Y).
  - Isto é equivalente à seguinte frase em lógica  
$$\forall x (\exists y \text{ Lecciona}(x,y)) \Rightarrow \text{Professor}(x)$$
- Sintaxe
  - Constantes começam com minúscula
  - Variáveis começam com maiúscula ou ‘\_’



# Mercury

- Linguagem de programação lógica/funcional desenvolvida na Universidade de Melbourne, Austrália
- Primeira versão em 1995
- Modular
- Tipagem forte
- Permite a definição de predicados através de factos e regras (*Horn clauses*) tal como no Prolog e com sintaxe idêntica
- Permite a definição de funções e tipos de dados como em ML, Caml, etc., embora com diferente sintaxe
- Permite a definição de predicados e funções de ordem superior
- Linguagem compilada, ao contrário do que acontece com outras linguagens declarativas

# Mercury (cont.)

- Ênfase na pureza declarativa
  - Do ponto de vista do paradigma lógico, é uma linguagem mais pura do que o Prolog
  - Tem pesquisa automática de alternativas (retrocesso), mas não suporta o corte de alternativas (*cut*)
  - Algumas otimizações possibilitadas pela pureza da linguagem permitem maior rapidez de execução