UNIVERSIDAD DE GRANADA

Aplicación de la teoría de tipos en el diseño de un lenguaje de programación orientado a la inteligencia artificial e implementación de su compilador

Doble Grado en Ingeniería Informática y Matemáticas

Bruno Santidrián Manzanedo

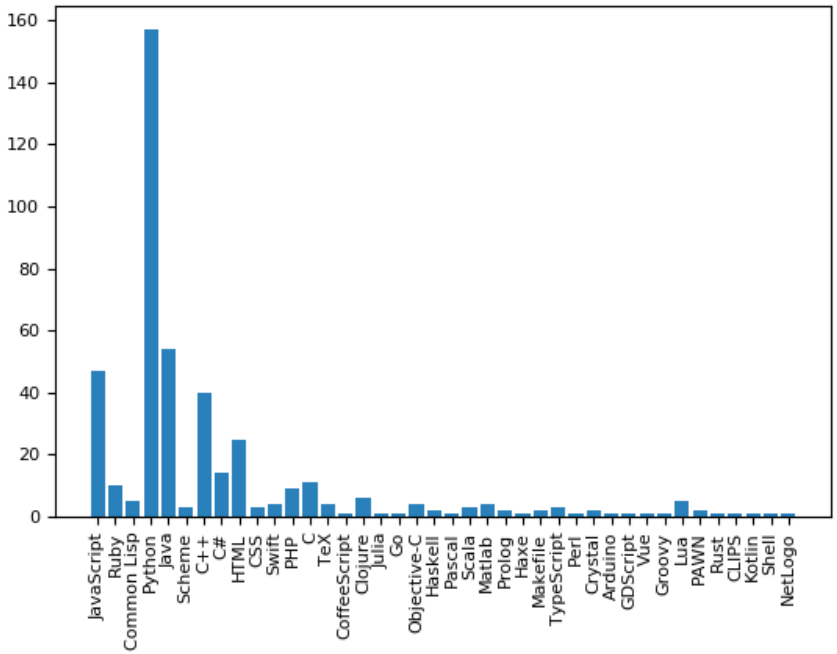
Índice

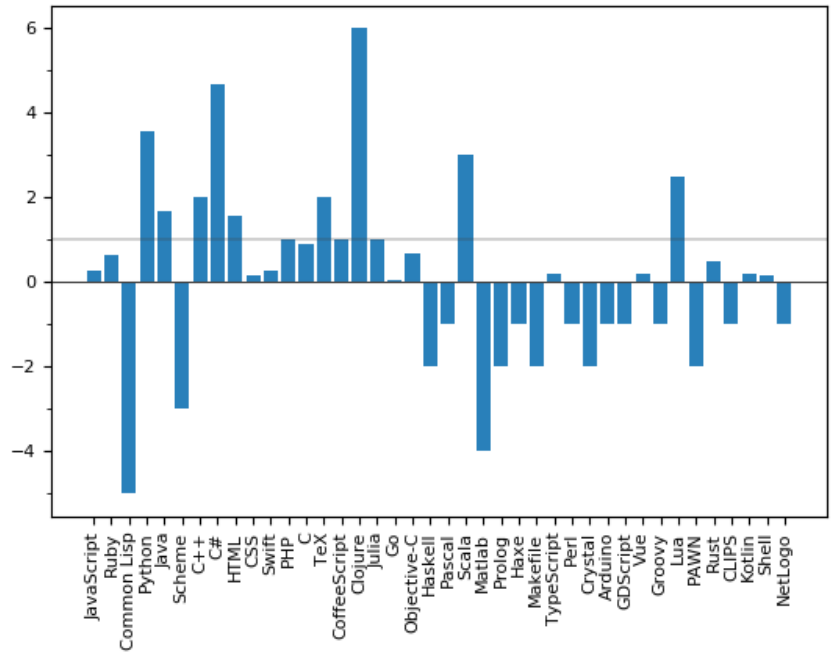
1. Motivación
2. Objetivos
3. Fundamentos matemáticos
4. Propuesta de lenguaje: *tail*
5. Conclusiones y trabajo futuro

Índice

1. Motivación
2. Objetivos
3. Fundamentos matemáticos
4. Propuesta de lenguaje: *tail*
5. Conclusiones y trabajo futuro

¿Hay espacio para un lenguaje especializado en IA?





¿Las diferencias son debidas al lenguaje o a las herramientas?

Elementos de un lenguaje especializado en IA

- Compilación AOT
- Recolección de basura
- Tipos graduales
- Concurrencia mediante paso de mensajes
- Facilidades a la computación simbólica
- Facilidades al álgebra lineal

Índice

1. Motivación
2. Objetivos
3. Fundamentos matemáticos
4. Propuesta de lenguaje: *tail*
5. Conclusiones y trabajo futuro

Objetivos

- Introducir la teoría de lenguajes de programación
- Relacionar la teoría de tipos y los sistemas de tipos
- Diseñar un lenguaje especializado en inteligencia artificial
- Implementar el compilador de dicho lenguaje

Índice

1. Motivación
2. Objetivos
3. Fundamentos matemáticos
4. Propuesta de lenguaje: *tail*
5. Conclusiones y trabajo futuro

Teoría de lenguajes de programación

Rama de las ciencias de la computación que estudia distintos aspectos de los lenguajes de programación

- Cómo diseñar lenguajes
- Cómo formalizar lenguajes
- Cómo implementar lenguajes
- Se apoya en teorías matemáticas como el calculo lambda, la teoría de tipos o la teoría de categorías

Teoría de tipos

Formalismo matemático que puede servir como fundamentación de las matemáticas

- Todo es un tipo, menos los términos

$$a : A \quad \begin{cases} a \text{ es evidencia de } A \\ a \in A \end{cases}$$

- No existe el conjunto de todos los conjuntos. Se utiliza una torre de universos $U_1 : U_2 : \ldots : U_i : \ldots$

$$\text{Ya no tiene sentido plantear } A = \{X \mid X \notin X\}$$

- Tipo $A \to B$

Se construye como $\lambda(x : A).\Phi \ : A \to B$ si $\Phi : B$ cuando $x : A$

Se le da el nombre $f : A \to B$ con $f :\equiv \lambda(x : A).\Phi$

Se utiliza con $f(a)$ si $a : A$

- Tipo $A \times B$

Se construye como $(a, b) : A \times B$ si $a : A$ y $b : B$

Se utiliza con $f((a, b)) :\equiv (g(a))(b)$ dado un $g : A \to B \to C$

- Tipo $\Pi_{(x:A)}B(x)$

  Dados $A : U$ y $B : A \to U$ se construye $f : \Pi_{(x:A)}B(x)$

  con $f :\equiv \lambda(x : A).\Phi$ con la diferencia de que $f(a) : B(a)$

- Tipo $\Sigma_{(x:A)}B(x)$

  Dados $a : A$ y $b : B(a)$ se construye como $(a, b) : \Sigma_{(x:A)}B(x)$

Cálculo lambda

$$(\lambda x.xx)\lambda x.xz$$

$$\begin{array}{lcll} t & = & & \text{términos} \\ & & \mid x & \\ & & \mid \lambda x.t & \\ & & \mid t\ t & \\ v & = & & \text{valores} \\ & & \mid \lambda x.t & \end{array}$$

$$\frac{t_1 \to t_1'}{t_1\ t_2 \to t_1'\ t_2} \qquad \frac{t_2 \to t_2'}{v_1\ t_2 \to v_1\ t_2'}$$

$$\frac{}{(\lambda x.t_{12})v_2 \to [x \mapsto v_2]t_{12}}$$

$$(\lambda x.xx)(\lambda x.xz) \to (\lambda x.xz)(\lambda x.xz) \to (\lambda x.xz)z$$

Cálculo lambda simplemente tipado

$$true : Bool$$

$$(\lambda x : Bool.x)true$$

$$\begin{array}{lll} t & = & \text{términos} \\ & | & x \\ & | & \lambda x{:}\ T.t \\ & | & t\ t \\ v & = & \text{valores} \\ & | & \lambda x{:}\ T.t \\ T & = & \text{tipos} \\ & | & T \to T \\ \Gamma & = & \text{contexto} \\ & | & \varnothing \\ & | & \Gamma, x : T \end{array}$$

$$\text{T-Var:}\ \frac{x : T \in \Gamma}{\Gamma \vdash x : T}$$

$$\text{T-Abs:}\ \frac{\Gamma, x : T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x : T_1.t2\ :\ T_1 \to T_2}$$

$$\text{T-Ap:}\ \frac{\Gamma \vdash t_1 : T_{11} \to T_{12}\ \ \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1\ t_2 : T_{12}}$$

Seguridad = Progreso + Preservación

Seguridad

Un sistema de tipos es seguro (y por extensión el lenguaje que lo contiene) si cualquier término bien tipado nunca se queda atascado

Progreso

Si un término está bien tipado no se encuentra atascado

Preservación

Si un término está bien tipado, tras un paso de cálculo lo sigue estando

Funciones μ-recursivas y turing-completitud

- Funciones constantes

$$f : \mathbb{N}^k \to \mathbb{N}$$
$$f(x_1, ..., x_k) \mapsto n$$

- Función sucesor

$$S : \mathbb{N} \to \mathbb{N}$$
$$S(x) \mapsto x + 1$$

- Funciones proyección

$$P_i^k : \mathbb{N}^k \to \mathbb{N}$$
$$P_i^k(x_1, ..., x_k) \mapsto x_i$$

- El operador de composición ($\circ$), sobre una función $h(x_1, ..., x_m)$ y m funciones $g_i(x_1, ..., x_k)$

$$(h \circ (g_1, ..., g_m))(x_1, ..., x_k) = h(g_1(x_1, ..., x_k), ..., g_m(x_1, ..., x_k))$$

- El operador de recursión primitiva (ρ), sobre dos funciones $g(x_1, ..., x_k)$ y $h(y, z, x_1, ..., x_k)$

$$\rho(g, h)(0, x_1, ..., x_k) = g(x_1, ..., x_k)$$
$$\rho(g, h)(y + 1, x_1, ..., x_k) = h(y, \rho(g, h)(y, x_1, ..., x_k), x_1, ..., x_k)$$

- El operador de minimización (μ), sobre una función $f(y, x_1, ..., x_k)$

$$\mu(f)(x_1, ..., x_k) = z \Leftrightarrow \begin{cases} f(z, x_1, ..., x_k) = 0 \\ f(i, x_1, ..., x_k) > 0 \quad \forall\ 0 \leq i \leq z - 1 \end{cases}$$

Índice

1. Motivación
2. Objetivos
3. Fundamentos matemáticos
4. Propuesta de lenguaje: *tail*
5. Conclusiones y trabajo futuro

Propuesta de lenguaje: *tail*

The Artificial Intelligence Language

- Unión e intersección de tipos graduales
- Captura de variables no inicializadas
- Átomos
- Variants generalizadas
- Tuplas como argumento
- Sobrecarga de funciones y operadores
- Números complejos y racionales

Algunas características destacables

```
f : Bool, ? -> ? or Int
f(b, x) := if b then x else 1

f(True, "hola")
```

```
f : :Complejo or :Racional -> Complex or Rational
f(a) := if a = :complejo then 2 + 3i else 3//4

f(:compeljo) # Error en tiempo de compilación

x : Atom := :cualquier_atomo
```

Algunas características destacables

```
variant Point :: Point(x, y) : Real, Real

·+· (p1, p2) := Point::Point(p1.x + p2.x, p1.y + p2.y)
```

```
x := [1 2 3 | 4 5 6 || 7 8 9]

x := [ 1 2 3 |
     | 4 5 6 |
     | 7 8 9 ]
```

```
f(x) := x, x+1, x+2
g(a, b, c) := a + b + c

g(f(1))
```

Algunas características destacables

Unión e intersección de tipos graduales

Con tipado estático la relación de subtipado $\leq$ es la inclusión

$$\gamma : \mathit{GTypes} \to \mathcal{P}(\mathit{STypes})$$

$$\gamma(?) \mapsto \mathit{STypes} \qquad \gamma(\tau_1 \to \tau_2) \mapsto \{T_1 \to T_2 \mid T_i \in \gamma(\tau_i)\}$$
$$\gamma(B) \mapsto \{B\} \qquad \gamma(\tau_1 \text{ or } \tau_2) \mapsto \{T_1 \text{ or } T_2 \mid T_i \in \gamma(\tau_i)\}$$
$$\gamma(\mathit{Void}) \mapsto \{\mathit{Void}\} \qquad \gamma(\tau_1 \text{ and } \tau_2) \mapsto \{T_1 \text{ and } T_2 \mid T_i \in \gamma(\tau_i)\}$$
$$\gamma(U) \mapsto \{U\} \qquad \gamma(\text{not } T) \mapsto \{\text{not } T\}$$

Gradual Extrema

Para todo tipo gradual $\tau \in \mathit{GTypes}$ existen dos tipos estáticos $\tau^{\Uparrow}$ y $\tau^{\Downarrow}$ tal que para todo $T \in \gamma(\tau)$, $\tau^{\Downarrow} \leq T \leq \tau^{\Uparrow}$

Algunas características destacables

Unión e intersección de tipos graduales

Extensión de la relación de subtipado

Para cada par σ, τ de tipos graduales definimos la relación $\widetilde{\leq}$ como:

$$\sigma \widetilde{\leq} \tau \Leftrightarrow \sigma^{\Downarrow} \leq \tau^{\Uparrow}$$

Para cada par σ, τ de tipos graduales definimos la relación $\widetilde{\not\leq}$ como:

$$\sigma \widetilde{\not\leq} \tau \Leftrightarrow \sigma^{\Uparrow} \not\leq \tau^{\Downarrow}$$

Algunas características destacables

Propiedades del lenguaje

Se puede demostrar que tail es turing-completo usando funciones μ-recursivas

Se puede demostrar que tail es un lenguaje seguro si todos sus términos están tipados de forma estática

Ejemplos de ejecución

```
x : Int


write("fib(x) = {fib(x)}")
```

```
x : (:A1 or :A2 ) and
    (:A2 or :A3 )

x := :a3
```

```
Semantic error:
In line 10, column 22

        write("fib(x) = {fib(x)}")
                         ^

The variable x is not initialized.
```

```
Semantic error:
In line 2, column 1

        x := :a3
        ^~~~~~~~

x is said to be of type (:A1 or :A2) and
(:A2 or :A3), but it's being assigned to
a type :A3.
```

Índice

1. Motivación
2. Objetivos
3. Fundamentos matemáticos
4. Propuesta de lenguaje: *tail*
5. Conclusiones y trabajo futuro

Conclusiones y trabajo futuro

Conclusiones

- Hemos presentado la teoría de lenguajes de programación
- Hemos estudiado la aplicación práctica de la teoría de tipos
- Hemos diseñado un lenguaje especializado en inteligencia artificial
- Hemos aplicado los resultados teóricos estudiados al análisis de nuestro lenguaje
- Hemos implementado parcialmente su compilador

Conclusiones y trabajo futuro

Trabajo futuro

- Completar la fase de generación de código
- Diseñar un sistema de módulos para tail
- Implementar un modelo de programación concurrente
- Estudiar en profundidad los tipos dependientes

¿Alguna pregunta?