Laboratorio de Lenguajes de Programación

Sistema verificador de tipos

Manuel Soto Romero

Universidad Nacional Autónoma de México
Facultad de Ciencias

9 de noviembre de 2016

Definición de tipo

Definición de tipo

¿Qué es un tipo?

Definición de tipo

¿Qué es un tipo?

¡Hablando de lenguajes de programación!

Definición de tipo

¿Qué es un tipo?

¡Hablando de lenguajes de programación!

Definición. (Tipo)

Convencionalmente se usa el término tipo para referirnos a una **abstracción de un conjunto de valores**. Un tipo etiqueta toda expresión en el lenguaje, marcando en qué clase de valor resultará evaluar dicha expresión.

Definición de tipo

¿Qué es un tipo?

¡Hablando de lenguajes de programación!

Definición. (Tipo)

Convencionalmente se usa el término tipo para referirnos a una **abstracción de un conjunto de valores**. Un tipo etiqueta toda expresión en el lenguaje, marcando en qué clase de valor resultará evaluar dicha expresión.

Los tipos describen invariantes que se mantienen en todas las ejecuciones de un programa.

Sistema verificador de tipos

Sistema verificador de tipos

¿De qué me sirve tener tipos en mi lenguaje?

Sistema verificador de tipos

¿De qué me sirve tener tipos en mi lenguaje?

Los tipos dan lugar a detectar errores en un programa mediante distintos mecanismos.

Sistema verificador de tipos

¿De qué me sirve tener tipos en mi lenguaje?

Los tipos dan lugar a detectar errores en un programa mediante distintos mecanismos.

Un sistema verificador de tipos es un conjunto de reglas que etiquetan toda expresión del lenguaje en una cierta abstracción de los valores que existen.

Sistema verificador de tipos

¿De qué me sirve tener tipos en mi lenguaje?

Los tipos dan lugar a detectar errores en un programa mediante distintos mecanismos.

Un sistema verificador de tipos es un conjunto de reglas que etiquetan toda expresión del lenguaje en una cierta abstracción de los valores que existen.

Un ejemplo clásico, es el sistema verificador de tipos de OCaml (descendiente de ML) que es bastante poderoso.

Sistema verificador de tipos

¿De qué me sirve tener tipos en mi lenguaje?

Los tipos dan lugar a detectar errores en un programa mediante distintos mecanismos.

Un sistema verificador de tipos es un conjunto de reglas que etiquetan toda expresión del lenguaje en una cierta abstracción de los valores que existen.

Un ejemplo clásico, es el sistema verificador de tipos de OCaml (descendiente de ML) que es bastante poderoso.

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

Sistema verificador de tipos

¿De qué me sirve tener tipos en mi lenguaje?

Los tipos dan lugar a detectar errores en un programa mediante distintos mecanismos.

Un sistema verificador de tipos es un conjunto de reglas que etiquetan toda expresión del lenguaje en una cierta abstracción de los valores que existen.

Un ejemplo clásico, es el sistema verificador de tipos de OCaml (descendiente de ML) que es bastante poderoso.

```
> let rec fact n = if (n = 0) then 1 else (n * (fact (n-1)));;
val fact : int -> int = <fun>
```

Sistema verificador de tipos

¿De qué me sirve tener tipos en mi lenguaje?

Los tipos dan lugar a detectar errores en un programa mediante distintos mecanismos.

Un sistema verificador de tipos es un conjunto de reglas que etiquetan toda expresión del lenguaje en una cierta abstracción de los valores que existen.

Un ejemplo clásico, es el sistema verificador de tipos de OCaml (descendiente de ML) que es bastante poderoso.

```
> let rec fact n = if (n = 0) then 1 else (n * (fact (n-1)));;
val fact : int -> int = <fun>
> fact 5;;
```

Sistema verificador de tipos

¿De qué me sirve tener tipos en mi lenguaje?

Los tipos dan lugar a detectar errores en un programa mediante distintos mecanismos.

Un sistema verificador de tipos es un conjunto de reglas que etiquetan toda expresión del lenguaje en una cierta abstracción de los valores que existen.

Un ejemplo clásico, es el sistema verificador de tipos de OCaml (descendiente de ML) que es bastante poderoso.

```
> let rec fact n = if (n = 0) then 1 else (n * (fact (n-1)));;
val fact : int -> int = <fun>
> fact 5;;
- : int = 120
```

Sistema verificador de tipos

¿De qué me sirve tener tipos en mi lenguaje?

Los tipos dan lugar a detectar errores en un programa mediante distintos mecanismos.

Un sistema verificador de tipos es un conjunto de reglas que etiquetan toda expresión del lenguaje en una cierta abstracción de los valores que existen.

Un ejemplo clásico, es el sistema verificador de tipos de OCaml (descendiente de ML) que es bastante poderoso.

```
> let rec fact n = if (n = 0) then 1 else (n * (fact (n-1)));;
val fact : int -> int = <fun>
> fact 5;;
- : int = 120
> fact [1;2];;
```

Sistema verificador de tipos

¿De qué me sirve tener tipos en mi lenguaje?

Los tipos dan lugar a detectar errores en un programa mediante distintos mecanismos.

Un sistema verificador de tipos es un conjunto de reglas que etiquetan toda expresión del lenguaje en una cierta abstracción de los valores que existen.

Un ejemplo clásico, es el sistema verificador de tipos de OCaml (descendiente de ML) que es bastante poderoso.

```
> let rec fact n = if (n = 0) then 1 else (n * (fact (n-1)));;
val fact : int -> int = <fun>
> fact 5;;
- : int = 120
> fact [1;2];;
Error: This expression has type 'a list
but an expression was expected of type int
```

Algunas ventajas de un sistema verificador de tipos

Algunas ventajas de un sistema verificador de tipos

- Ayudan a reducir el tiempo en depuración de un programa.

Algunas ventajas de un sistema verificador de tipos

- Ayudan a reducir el tiempo en depuración de un programa.
- Un sistema de tipos detecta errores en un código que no ha sido ejecutado por el programador, es decir, en tiempo de compilación.

Algunas ventajas de un sistema verificador de tipos

- Ayudan a reducir el tiempo en depuración de un programa.
- Un sistema de tipos detecta errores en un código que no ha sido ejecutado por el programador, es decir, en tiempo de compilación.
- Ayudan a documentar el programa.

RCFAE con tipos

RCFAE con tipos

Modificaremos la gramática de nuestro ”chiqui” lenguaje para que tenga tipos.

RCFAE con tipos

Modificaremos la gramática de nuestro ”chiqui” lenguaje para que tenga tipos.

```
<expr> ::= <id>
         | <num>
         | {binop <expr> <expr>}
         | {with <id> <expr> <expr>}
         | {rec <id> <expr> <expr>}
         | {fun <id> <expr>}
         | {<expr> <expr>}
```

RCFAE con tipos

Modificaremos la gramática de nuestro ”chiqui” lenguaje para que tenga tipos.

```
<expr> ::= <id>
         | <num>
         | {binop <expr> <expr>}
         | {with <id> <expr> <expr>}
         | {rec <id> <expr> <expr>}
         | {fun <id> <expr>}
         | {<expr> <expr>}
```

```
<expr> ::= <id>
         | <num>
         | {binop <expr> <expr>}
         | {with {<id> : <type>} <expr> <expr>}
         | {rec {<id> : <type>} <expr> <expr>}
         | {fun {<id> : <type>} : <type> <expr>}
         | {<expr> <expr>}
```

La gramática type

La gramática type

```
<type> ::= number
         | (<type> -> <type>)
```

La gramática type

```
<type> ::= number
         | (<type> -> <type>)
```

Ejemplos:

La gramática type

```
<type> ::= number
         | (<type> -> <type>)
```

Ejemplos:

```
{fun {n : number} : number {pow n 2}}
```

La gramática type

```
<type> ::= number
         | (<type> -> <type>)
```

Ejemplos:

```
{fun {n : number} : number {pow n 2}}
```

```
{rec {fac {fun {n : number} : number
         {if0 n 1 {* n {fact {- n 1}}}}}}}
   {fac 5}}
```

La gramática type

```
<type> ::= number
         | (<type> -> <type>)
```

Ejemplos:

```
{fun {n : number} : number {pow n 2}}
```

```
{rec {fac {fun {n : number} : number
         {if0 n 1 {* n {fact {- n 1}}}}}}
   {fac 5}}
```

¿Qué debería hacer un sistema verificador de tipos con la siguiente expresión?

```
{{fun {n : number} : number {pow n 2}}
   {fun {x : number} : number {+ x x}}}
```

El procedimiento typeof

El procedimiento typeof

El procedimiento typeof será el encargado de implementar nuestro sistema verificador de tipos. Éste recibe una expresión en sintaxis abstracta, por lo que tenemos que acoplar nuestro parser.

El procedimiento typeof

El procedimiento typeof será el encargado de implementar nuestro sistema verificador de tipos. Éste recibe una expresión en sintaxis abstracta, por lo que tenemos que acoplar nuestro parser.

Ejemplos:

El procedimiento typeof

El procedimiento typeof será el encargado de implementar nuestro sistema verificador de tipos. Éste recibe una expresión en sintaxis abstracta, por lo que tenemos que acoplar nuestro parser.

Ejemplos:

```
> (typeof (num 2))
(tnumber)
```

El procedimiento typeof

El procedimiento typeof será el encargado de implementar nuestro sistema verificador de tipos. Éste recibe una expresión en sintaxis abstracta, por lo que tenemos que acoplar nuestro parser.

Ejemplos:

```
> (typeof (num 2))
(tnumber)
```

```
> (typeof (binop + (num 2) (fun 'x (tnumber) (tnumber) (id 'x))))
"Error: En + se esperaban argumentos de tipo number
```

El procedimiento typeof

El procedimiento typeof será el encargado de implementar nuestro sistema verificador de tipos. Éste recibe una expresión en sintaxis abstracta, por lo que tenemos que acoplar nuestro parser.

Ejemplos:

```
> (typeof (num 2))
(tnumber)
```

```
> (typeof (binop + (num 2) (fun 'x (tnumber) (tnumber) (id 'x))))
``Error: En + se esperaban argumentos de tipo number
```

Para implementar el procedimiento typeof se sugiere usar un ambiente. Funcionará como el de nuestro intérprete sólo que los resultados serán los tipos de cada expresión.

El procedimiento typeof

El procedimiento typeof será el encargado de implementar nuestro sistema verificador de tipos. Éste recibe una expresión en sintaxis abstracta, por lo que tenemos que acoplar nuestro parser.

Ejemplos:

```
> (typeof (num 2))
(tnumber)
```

```
> (typeof (binop + (num 2) (fun 'x (tnumber) (tnumber) (id 'x))))
``Error: En + se esperaban argumentos de tipo number
```

Para implementar el procedimiento typeof se sugiere usar un ambiente. Funcionará como el de nuestro intérprete sólo que los resultados serán los tipos de cada expresión.

identificador - tipo