



Auxiliar 4

Parsing e Intérprete

P1. El archivo `base_aux4.rkt` contiene una definición inicial del tipo de datos recursivo `Expr` con constantes numéricas, adición y resta. Además contiene su parser e interpretación. Nuestro objetivo es construir una super calculadora que soporte operaciones lógicas.

a) Extienda `Expr` y el `parse` con nuevas operaciones aritméticas, booleanas y de control de flujo, y complete el parser. En específico agregue los siguientes constructores:

- Multiplicación y división.

```
>>> (parse '(* 1 2))      >>> (parse '(/ 1 2))
(mul (num 1) (num 2))    (div (num 1) (num 2))
```

- Booleanos.
- Las operaciones lógicas `and`, `or` y `not`.

```
>>> (parse '(and #t #f))  >>> (parse '(or #t #f))
(land (bool #t) (bool #f)) (lor (bool #t) (bool #f))
```

```
>>> (parse '(not #t))
(neg (bool #t))
```

- Operación menor-que.

```
>>> (parse '(< 1 2))
(less (num 1) (num 2))
```

- Expresión `if` donde la primera expresión evalúa a un booleano, y la segunda y tercera a ramas `true` y `false`.

```
>>> (parse '(if (not #t) 1 2))
(ifp (neg #t) (num 1) (num 2))
```

b) Complete `calc` con las nuevas operaciones. Maneje el caso de división por 0. Por ahora asuma que el programa está bien escrito y no hay conflictos de tipos.

P2. *Chequeo estático de tipos:* Como no podemos confiar en nadie y si hay problemas de tipos nuestra calculadora va a crashear, es necesario evitar calcular un programa mal tipado. Para esto:

- a) Defina el tipo `Type` con los tipos de los resultados posibles de nuestro programa.
- b) Implemente la función `typecheck` de tipo `Expr -> Type/error` que devuelve el tipo de un programa bien tipado, o, en caso contrario, arroja un error de tipo. Los errores deben tener este formato:

```
"Static type error: expected T1 found T2"
```

Considere que:

- Las operaciones booleanas reciben y devuelven booleanos.
 - Comparadores reciben números y devuelven booleanos.
 - Ambas ramas del `if` deben tener el mismo tipo.
- c) Para terminar y juntar todo defina la función `run` de tipo `s-expr -> Val/error`, que aplique `parse`, `typecheck` y `calc` sobre un programa y retorne su valor final o lance un error de tipo.

```
>>> (run '(if (not #t) 1 2))  
2
```

```
>>> (run '(if (* 2 3) 1 2))  
"Static type error: expected Bool found Num"
```

P3. *Propuesto:* Extienda el lenguaje para que soporte pares y sus operaciones `fst` y `snd`.

```
<type> ::= Num | Bool | {Pair <type> <type>}
```

```
<expr> ::= ...  
         | {fst <expr>}  
         | {snd <expr>}
```