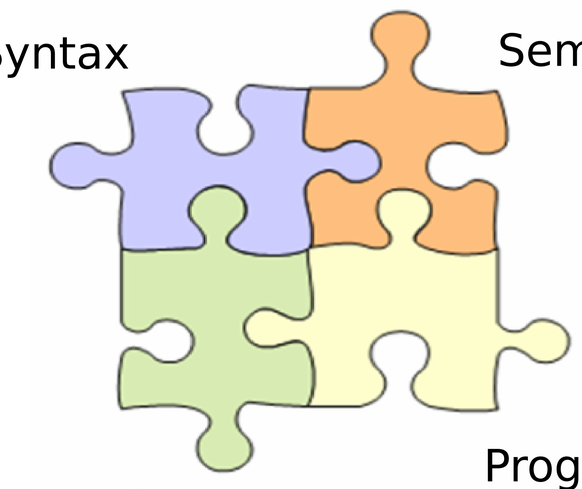
Extensible type systems

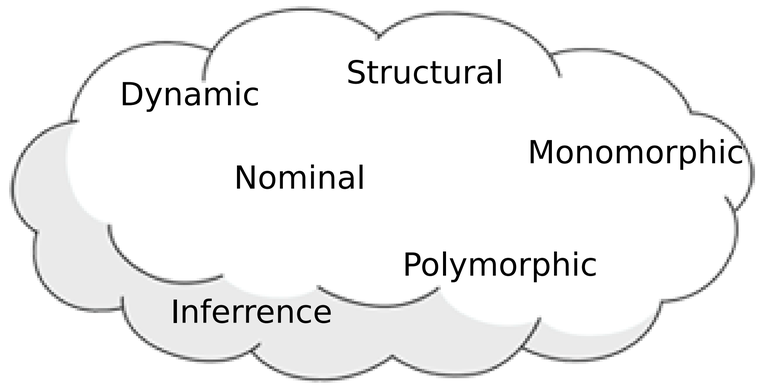
Jon Rafkind

Matthew Flatt



Domain specific languages can be embedded inside a host language. Designers of new languages should have the ability to make different design choices than the host language. The most common form of extensibility in modern languages is syntactic extension but we claim that extensible type systems play another key role in the design space.

What type system works best? Which type systems do programmers want to use?



Type system design choices

Languages can be defined almost entirely with macros. PLT Scheme macros are powerful enough to let the programmer define his own type system.

Macros are compile time user defined functions

```
(define-syntax (for-loop syntax)
  (syntax-case syntax (in)
    [(_ iterator in values body ...)
     #'(let loop ([rest values])
         (cond
           [(null? rest) (void)]
           [else (let ([iterator (car rest)])
                   body ... (loop (cdr rest)))]))]))
```

```
(for i in (list 1 2 3)
   (do-work (+ i 1))
```

⇓

```
(let loop ([rest (list 1 2 3)])
  (cond
     [(null? rest) (void)]
     [else (let ([i (car rest)])
             (do-work (+ i 1))
              (loop (cdr rest)))]))
```

Define the typing rules for a given expression along with a macro.

```
(define-syntax/type (for-loop syntax)
  (syntax-case syntax (in)
    [(_ iterator in values body ...)
       (values
          (type-constraint (element-type
                                iterator
                                values))
          #'(let loop ([rest values])
              (cond
                 [(null? rest) (void)]
                 [else (let ([iterator (car rest)])
                         body ... (loop (cdr rest)))]))]))
```

```
#lang honu

obj add1(x){
  x + 1
}

(int -> int) get(){
   add1
}
```

```
#lang VHDL

library ieee;
use ieee.std_logic_1164.all;

----------------------------------------

entity NOR_ent is
port( x: in std_logic;
 y: in std_logic;
 F: out std_logic
);
end NOR_ent;
```