classmate

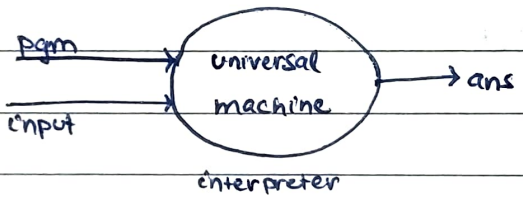
Date 29-7-2019

Page 1

INTRODUCTION

- what is a pgm?
- what is a PL?
- how does a pgm run? — abstract models of computing m/c
- how is a PL designed?

universal turing machine — what alan turing proposed?



what is the meaning of a program?

LISP
↑
Schemelang ← RACKET

reading assignments

Office hours (CERC) moodle atreyaa.ghosal@r.

3:30 Q1 mid Q2 end + assignments

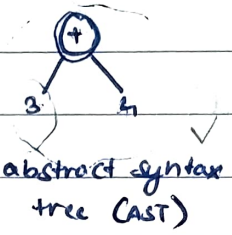
racketlang.org - guide: «reading»

emacs - org mode

1. programs are expressions,
that may be represented as trees.

3+4

concrete syntax



abstract syntax tree (AST)

pgm → parser → AST

AST ← interpreter

metal extraction — milling - product.

+ 3 4 — easiest

Prefix form

2. programs are evaluated by simplifying expressions.

design aesthetics

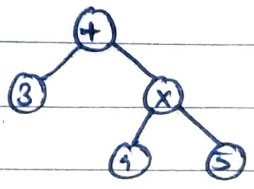
(3 + (4 x 5))

(+ 3 4) - 7

> (+3 (x 4 5))

(+ 3) - 3

(+) - 0



literate programming "essay" and inside code.

functional programming

classmate
Date 1.8.2019
Page 1

INTRODUCTION

EOPL essentials of programming languages
friedman, wand.

HtDP how to design programs
felleisen, findler, flatt, krishnamurthi.
course objectives htdp (1-8) step.
the racket guide (1-5) eopl (1)

programming languages: application and interpretation,

FUNCTIONAL PROGRAMMING

DATATYPES

- PRODUCTS $A \times B$ ← python tuples.
- DISJOINT UNIONS $A + B := \{red\} \times A \cup \{blue\} \times B$

$\{1, 2\} + \{1, 5\}$

tags ⟶ $\{(red, 1), (red, 2), (blue, 1), (blue, 5)\}$
class name ↗

function: $x \mapsto x+2$
- procedure
- table

(lambda (x) (* x 2))

(lambda (x)
  (* x 2))

```
(define add2
  (lambda (x)
    (* x 2)))

(add2 5)
10
```

TOPICS

0. ABSTRACT SYNTAX ← structure
1. SCOPE ← identifiers ← objects
2. STATE ← memory (store)
3. CONTROL ← exec context
4. TYPES
5. EFFECTS ← actions that change things

logic programming (Prolog) → 3. CONTROL
concurrency → 3. CONTROL

* is also a procedure.

5 → general purpose.

```
((lambda (*) (+ * 1)) 5)
```

↑ local identifier.

```
(let ((x 5))
  (+ x 1))
6
```

```
(let ((x e) ...) body)

((λ (x ...) body) e ...)
```

```
(define x 6)
(let ((x 2))
  (add1 x))
3

x
6
```

```
(define d/dx
  (lambda (f)
    (lambda (x)
      (/ (- (f (+ x dx))
            (f x))
         dx))))
```

lists ... recursion

emacs org-mode 9.x +

23.x +

classmate
Date 5-8-2019
Page 1

EMACS ORG MODE

literate programming

instead of comment — narrative

#+BEGIN_SRC python :tangle func.py

def function(n):

→ expandable for code highlighting

#+END_SRC ctrel-cvt

possible to tangle code blocks

in desired order using named blocks.