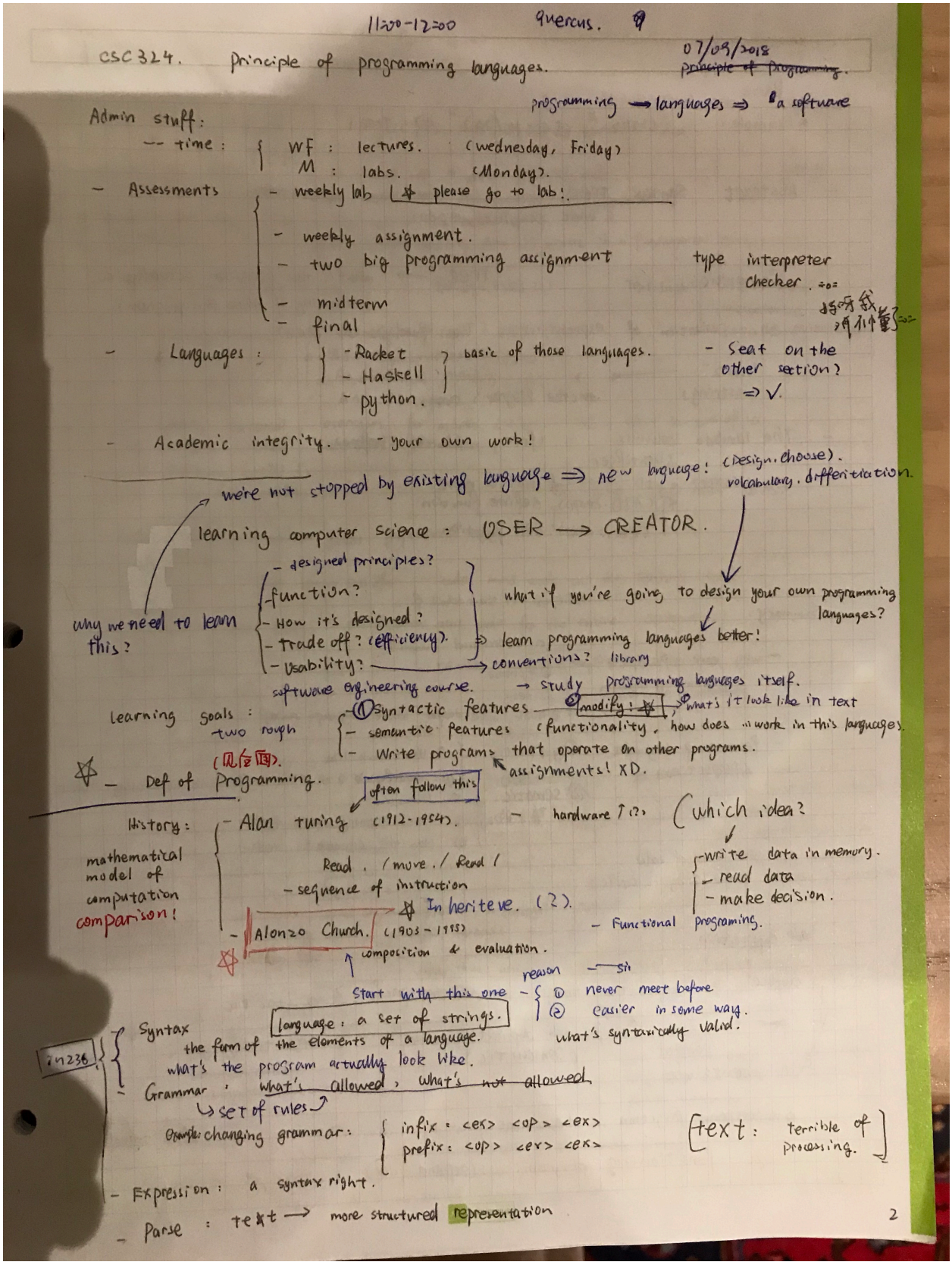
week0

Thursday, 13 September 2018 10:31 AM



Parse:

* Example: "(3+4) * (5 - (10/2))" ⇒ Tree!

- Abstract Syntax tree. (a tree-based representation). AST.
  (what compiler will do).

- Semantics. Tree → what's the program actually do?
  (meaning of ~~these~~ this program).

computation as Evaluation of expressions: (The fundamental idea).

"(1+2" (string) → 3 (the program's means).

- The lambda calculus. (from "Church) (before).
  IDentifier.

<expr> = ID → just a name
| (λ ID . <expr>) define function
| (<expr> <expr>) call function.

Summary.

- why learn Principle of ~~pro~~ programming languages
  ① understand, work↑.
  ② choose
  ③ design.
  ☆think more deeper above language itself

- Features of languages (properties)
  ① Syntactic (语法)
    ① How's look like
    ② modify. ☆☆ ⇒ cool!!!, that's why we're using racket.
  ② semantic. (functioning)
    ① How ...
    ② How's memory model work
    ③ How the argument work
    .....

- what kind of code you're going to write?

~~code~~ duplication

- surface level stylistic things. → white space... } syntactic
- ~~poor~~ some IDE (powerful) look for code duplication (semantic), refactor, automatically gap...
- ☆ Assignments! xD.

☆ understand programs in theoratical point of view.

☆ functioning program. (expand def)

CS 324 08/10/2018

History comparison:

① Alan Turing: computer scientist
- He developed a model of computation: Determine machine.

actually they're equivalent.

⇒ influence all of modern hardware.
Determine machine → A CPU style chip - make decision. (in CSC 236)
finite state machine.

① state machine + tape (∞ square, 0 or 1) ②
read/write 0/1

Introduction to the Theory of computation
a metaphore.

② influence most of modern programming languages
writing/reading from memory, making decisions.

sequence of instructions:
read, move, read → if [- → ..... →

"imperative" 祈使的, 命令的
telling someone what todo.

② Alonzo Church.
also a mathematical model of computation.
IDEA: To use [computation] to evaluate [computational functions]
Start with this. ☺ cool ☺ simpler.
f(x) = x+2, composition of functions.
composition & evaluation of functions
f(g(x)) .....
emphasize Ideal programming "functional programming".
like we can understand f(1)=3 : we substitute.
easier to work with comparing to move left, write... in (Turing)

In CSC 236: language → a set of strings.
SETS
Syntax & Grammar.
Syntax
Grammar.
look like
rules. (Grammar include Syntax).
string content.

Example: ① infix

<expr> = NUMBER | '(' <expr> <op> <expr> ')'
<op> = '+' | '-' | '*' | '/'

variables.
terminal symbols. ⇒ literals. (文字).
non-terminal symbols.

CAP:
" (": string
< >: variable.

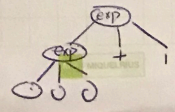
A grammar to generate valid arithmetic expressions

② prefix

<expr> = NUMBER | "(" <op> <expr> <expr> ')'
<op> = '+' | '-' | '*' | '/'

Expression = A syntactically-valid element of a language.
Parse (解析): text → "more structured" representation.
① Recursive
show ⇒ ② To follow the grammar = expression = 3 parts.

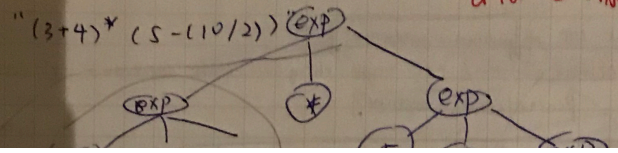
Tree: Abstract Syntax tree
A tree based representation of program (generated from code).



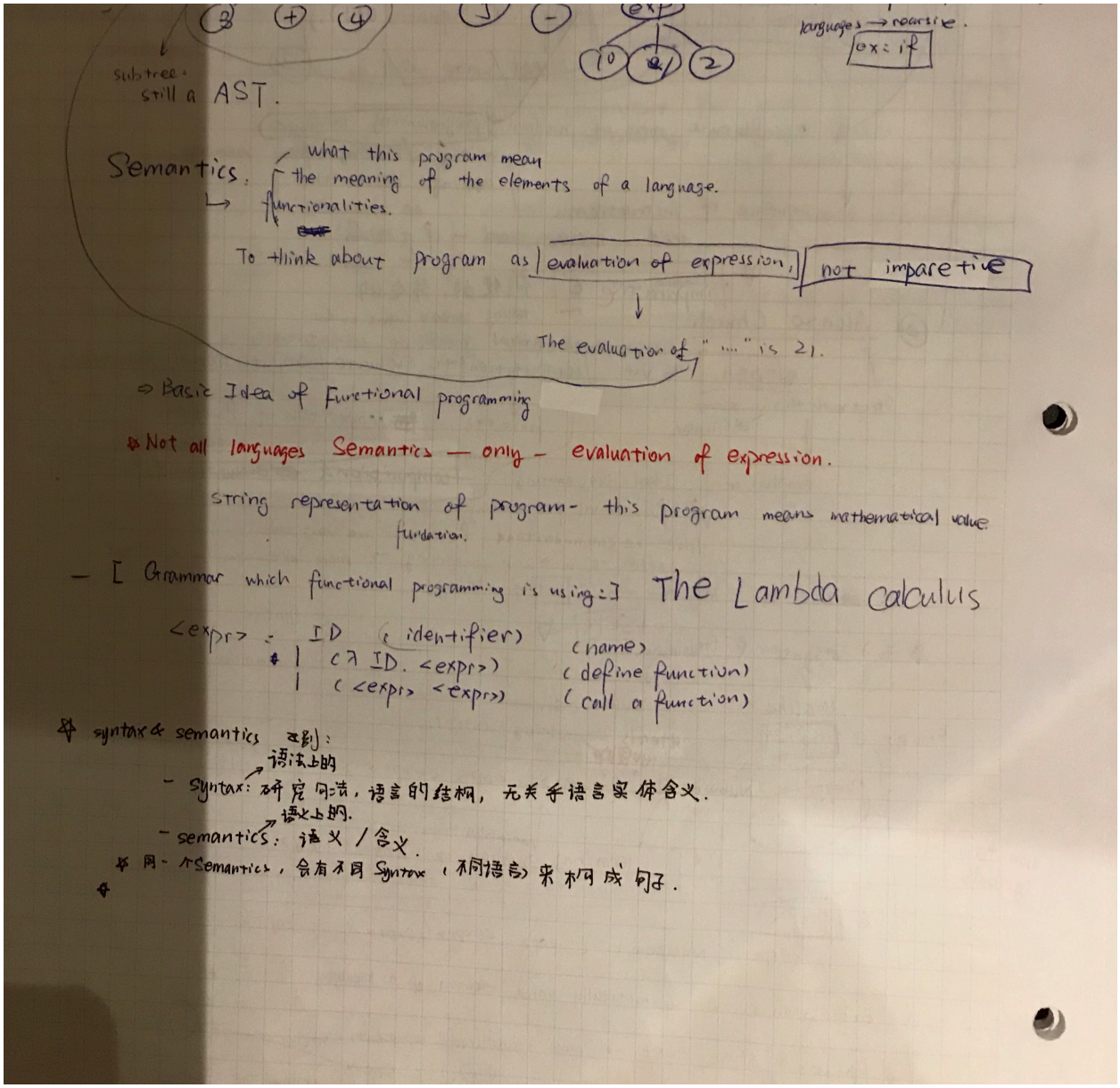
Q: ~~syntax~~ & semantics & grammar?

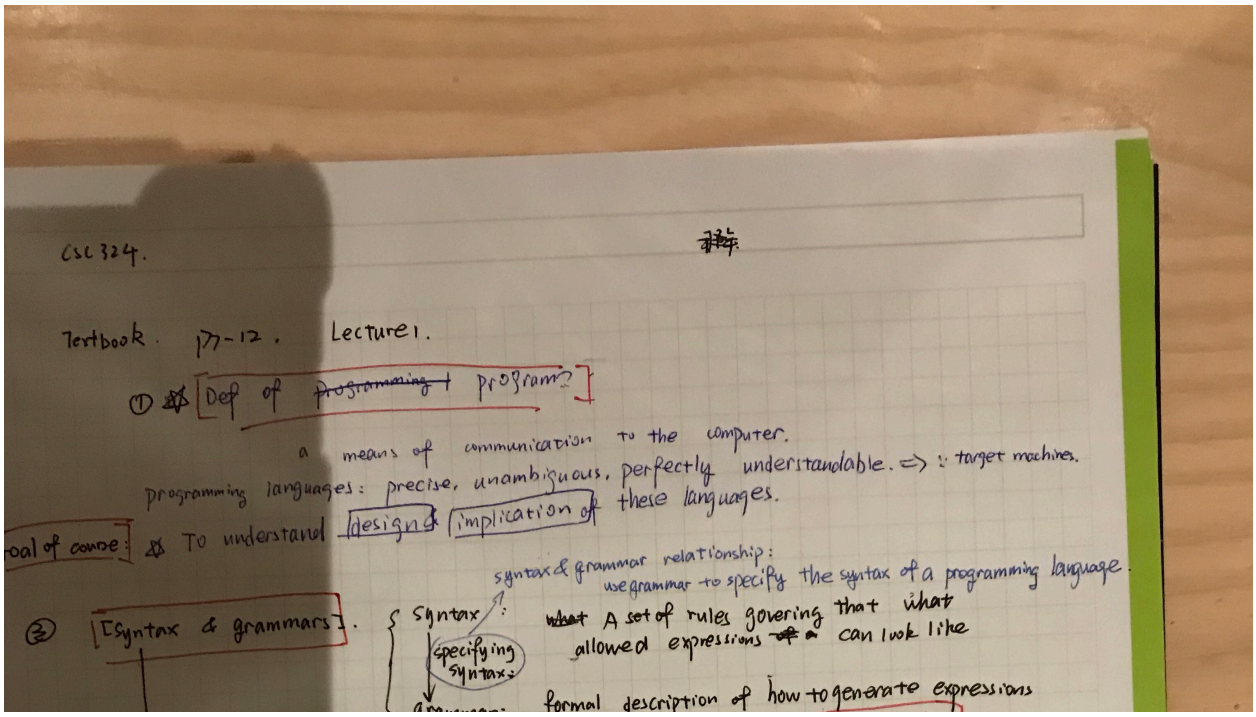
All tree structure! — internal nodes show the complex elements of the language
- leaves: ~~atomic~~ elements of language = literals, leaves
atomic (原子的)

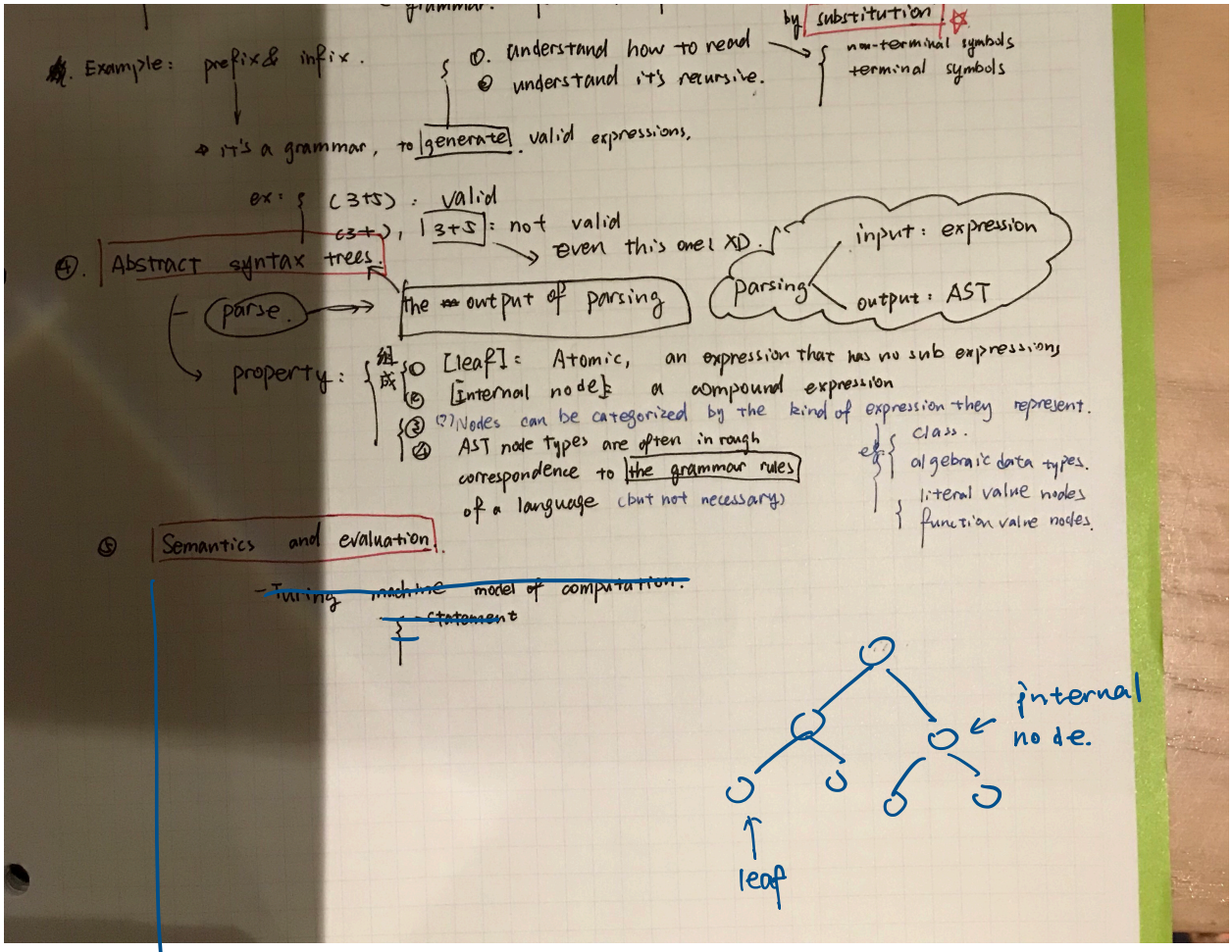
"(3+4)*(5-(10/2))"



Trees are recursive.
① Grammar rules - recursive
② Grammar for all programming


(3) (+) (4)
exp
10 2 2
languages → recursive.
ex: if
subtree:
still a AST.
Semantics: what this program mean
the meaning of the elements of a language.
functionalities.
To think about program as evaluation of expression; not imperative
The evaluation of " .... " is 21.
⇒ Basic Idea of Functional programming
* Not all languages Semantics — only — evaluation of expression.
string representation of program - this program means mathematical value
[Grammar which functional programming is using :] The Lambda calculus
<expr> ::= ID (identifier) (name)
| (λ ID. <expr>) (define function)
| (<expr> <expr>) (call a function)
* syntax & semantics 区别:
- Syntax: 研究句法，语言的结构，无关乎语言实体含义.
- semantics: 语义 / 含义.
* 同一个semantics，会有不同 Syntax (不同语言) 来构成句子.


CSC 324.
Textbook. 1.1-1.2. Lecture 1.
① Def of programming / program?
a means of communication to the computer.
programming languages: precise, unambiguous, perfectly understandable. ⇒ : target machines.
Goal of course: * To understand design & implication of these languages.
② [Syntax & grammars].
syntax & grammar relationship:
use grammar to specify the syntax of a programming language.
Syntax: A set of rules govering that what allowed expressions can look like
specifying syntax:
formal description of how to generate expressions



⑤ Semantics & evaluation

- imparative

① [Def of program.]

② [Goal of courses]

③ [Syntax & grammars]

④ [Astract syntart trees]

⑤ [Semantics & evaluation].