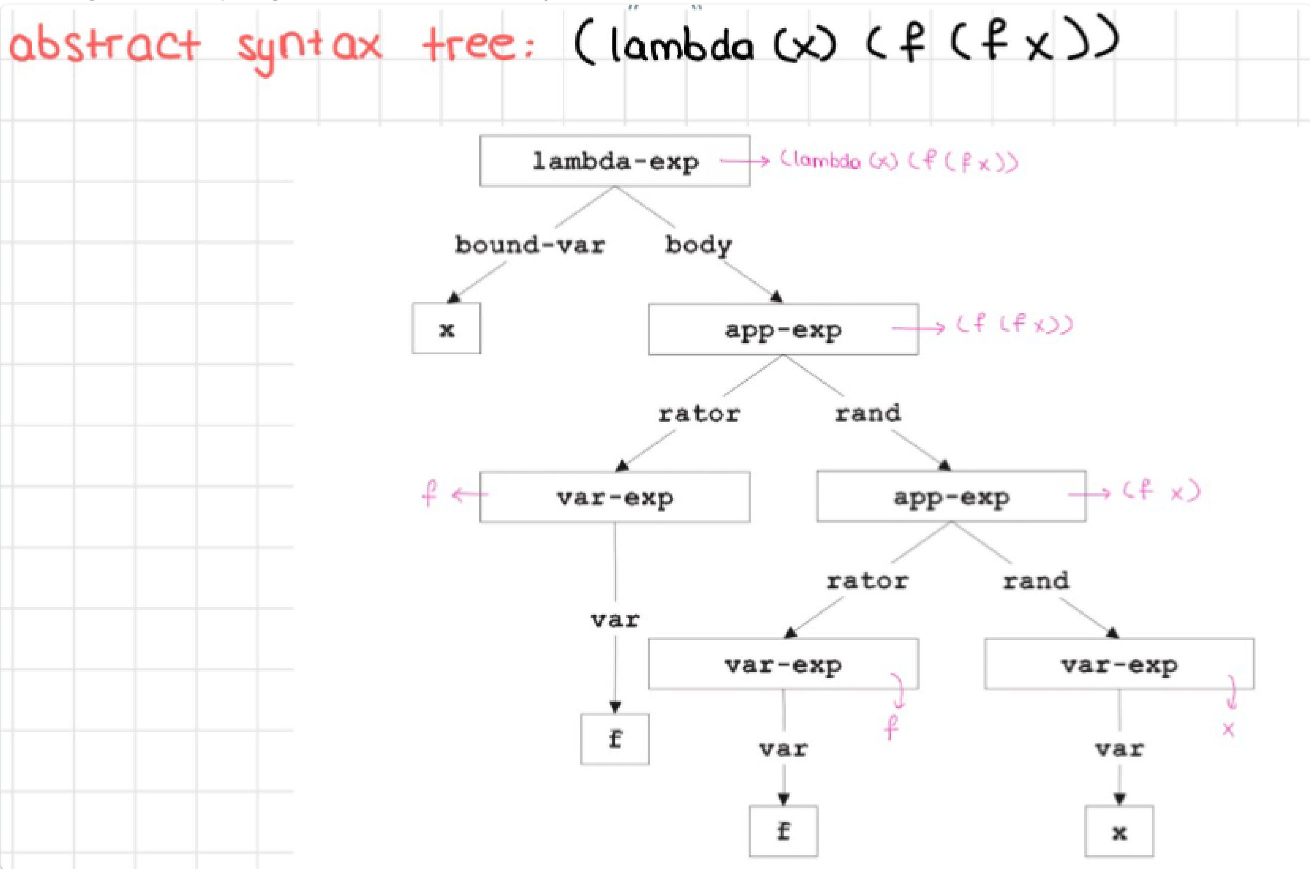
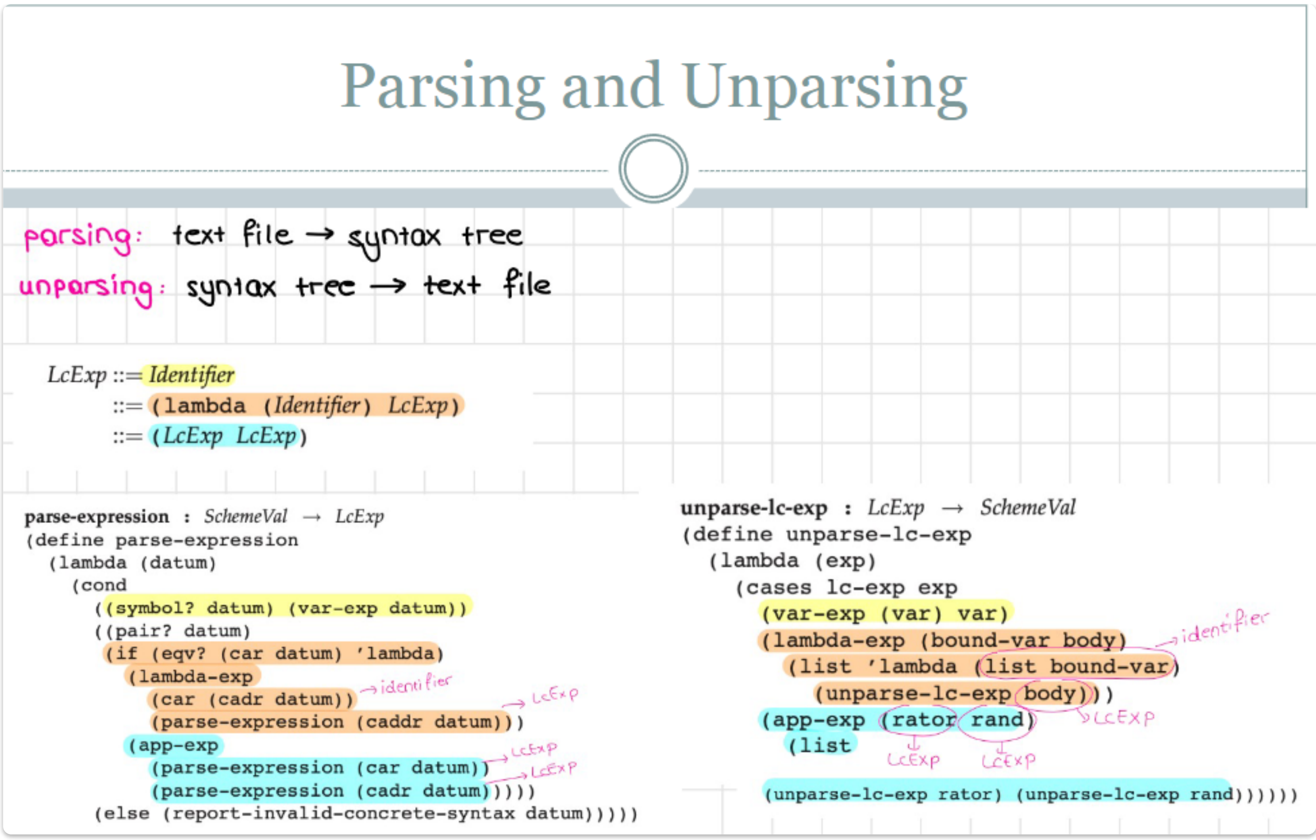
10. Abstract Syntax, Representation, Interpretation

[lecture 10 -- Syntax Representation & Interpreters.pdf]

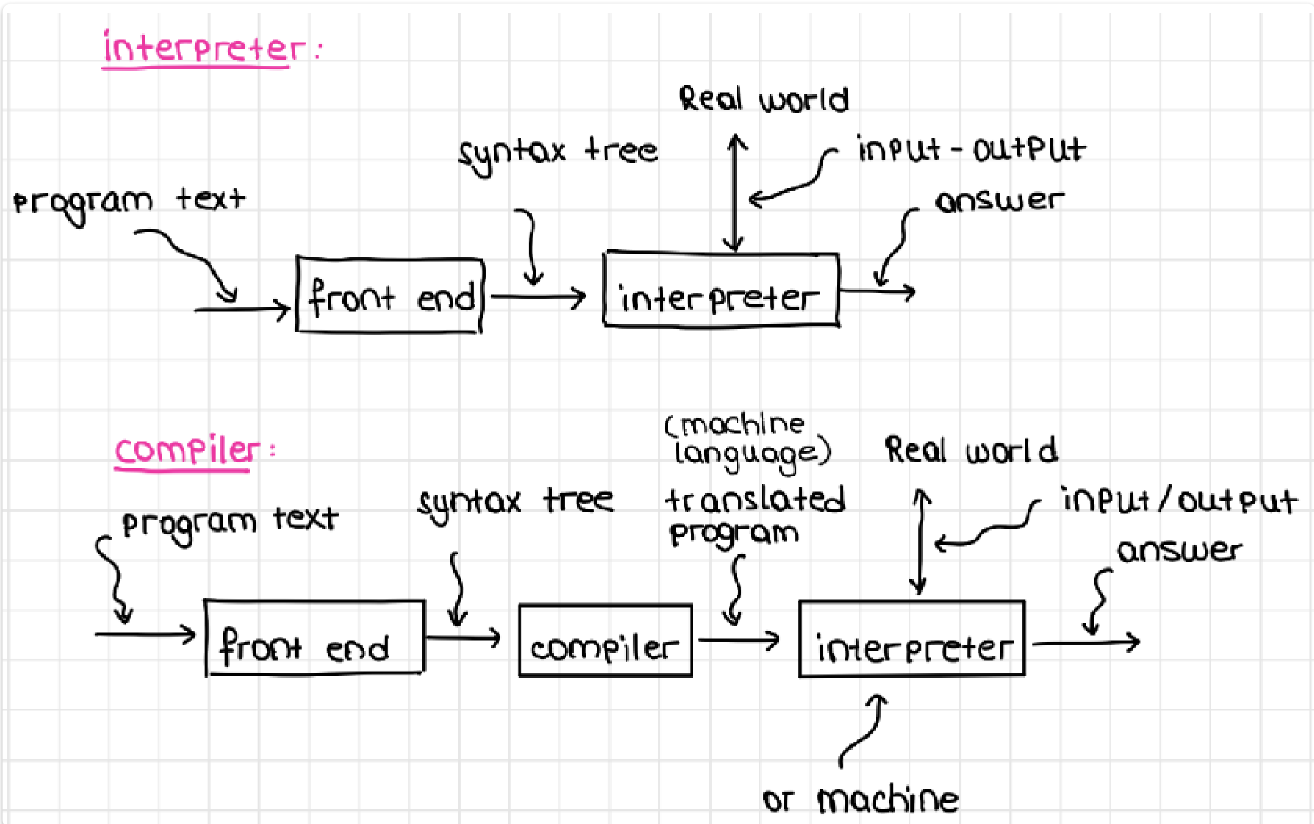
Abstract Syntax Tree

Parsing takes a program and builds a syntax tree.





Compilers



Let Language

LET Language:

grammar

Program ::= *Expression* → concrete syntax

`a-program (exp1)` → abstract syntax

Expression ::= *Number*

`const-exp (num)`

Expression ::= `-(`*Expression* `,` *Expression*`)`

`diff-exp (exp1 exp2)`

Expression ::= `zero?` `(`*Expression*`)`

`zero?-exp (exp1)`

Expression ::= `if` *Expression* `then` *Expression* `else` *Expression*

`if-exp (exp1 exp2 exp3)`

Expression ::= *Identifier*

`var-exp (var)`

x = 5 -(x,5)

Expression ::= `let` *Identifier* `=` *Expression* `in` *Expression*

`let-exp (var exp1 body)`