Standardizing RPAL AST's

Programming Languages
Lecture 6

Adeesha Wijayasiri

The Semantics of RPAL

We use the operational approach:

To specify the semantics of a language, we give an OPERATIONAL PROCESS.

PL constructs are "denoted" with functions whose behavior specify the meaning of the program.

The RPAL Operational Specification

- In the case of RPAL, the process is
  1. Scan and parse the program, transduce to a tree.
  2. Standardize the tree.

The RPAL Operational Specification (cont'd)

3. Flatten the tree into either:
   a. A lambda expression.
   b. Control Structures for a machine (more later)

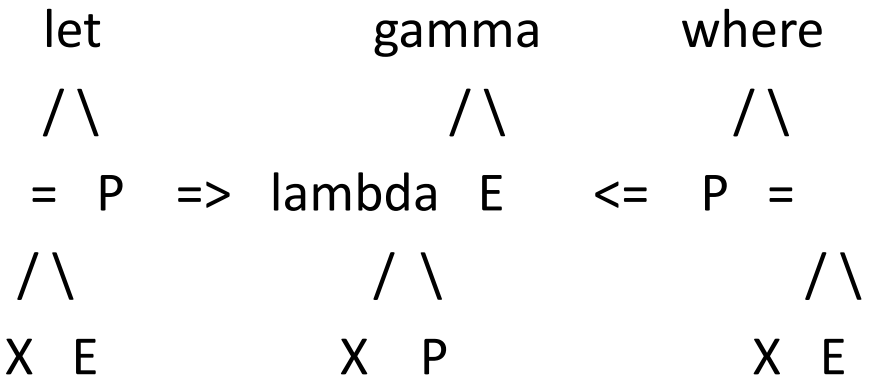
- Important:
  - RPAL = lambda-calculus + syntactic sugar

Standardizing an RPAL AST

- "Desugar" the tree.
- Transform into a binary tree, whose internal nodes are exclusively 'gamma' and 'lambda'.

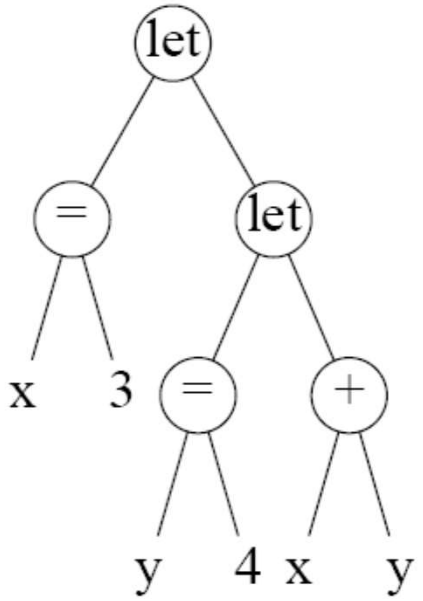
Standardizing 'let' and 'where'

- See example



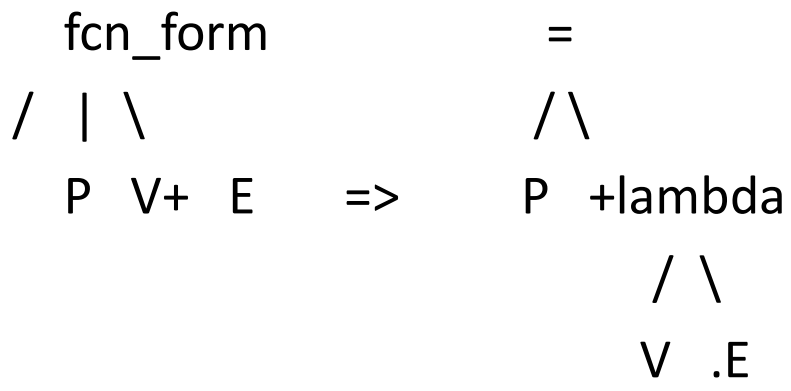
Standardizing 'let' and 'where':

```
let x=3 in let y=4 in x+y
```



Standardizing 'fcn_form'

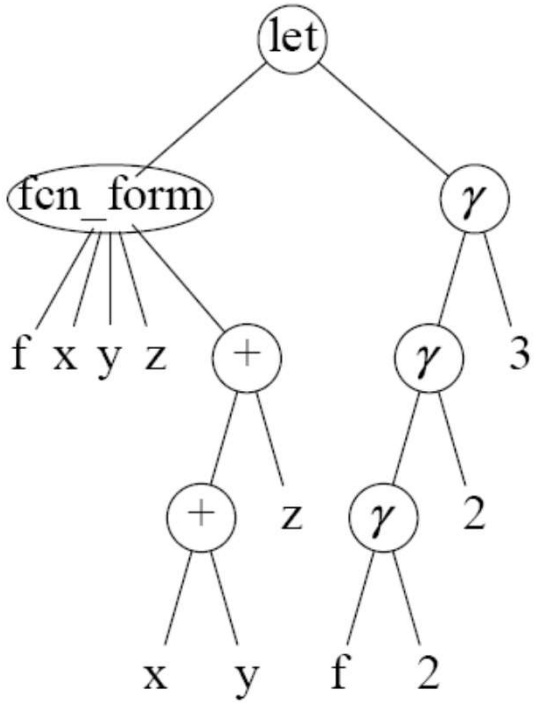
- See example



- V+ means "one or more", to be repeated likewise on right side.
- The 'dot' indicates the location of the pattern's repetition.

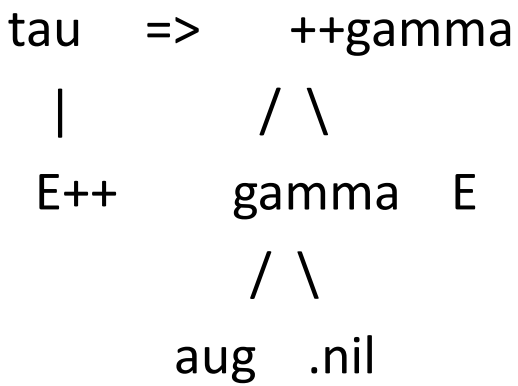
Standardizing 'fcn_form':

```
let f x y z = x+y+z in f 1 2 3
```



Standardizing Tuples

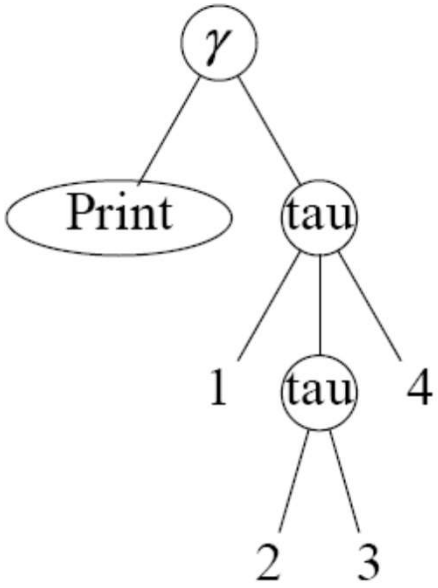
- See example



E++ means “two or more” E’s.

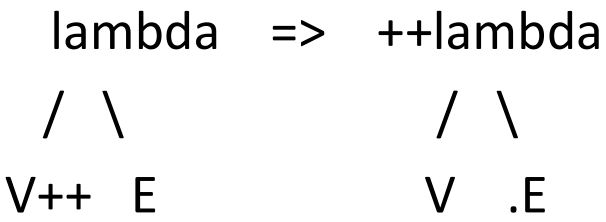
Standardizing tuples:

```
Print (1,(2,3),4)
```



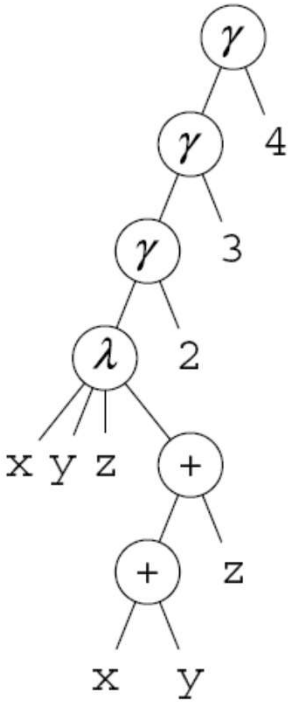
Standardizing Multi-Parameter Functions

- See example



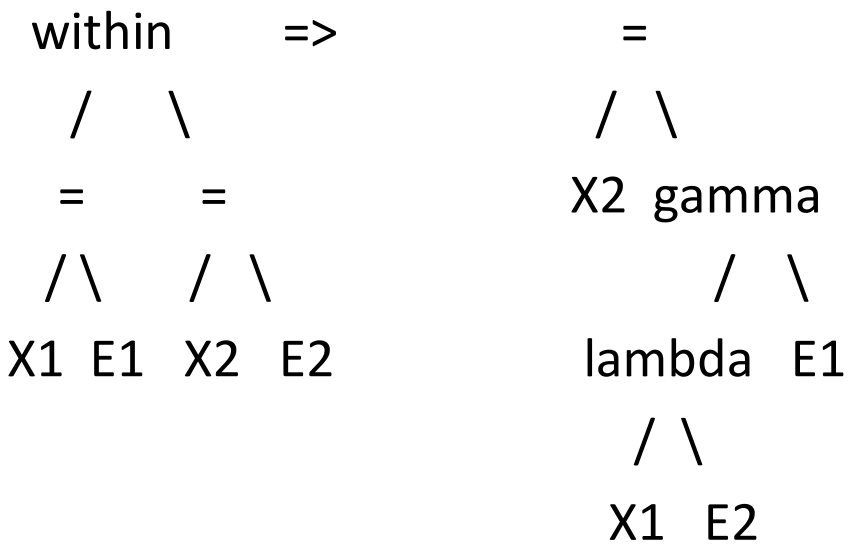
Standardizing multi-parameter functions:

```
(fn x y z. x + y + z) 2 3 4
```



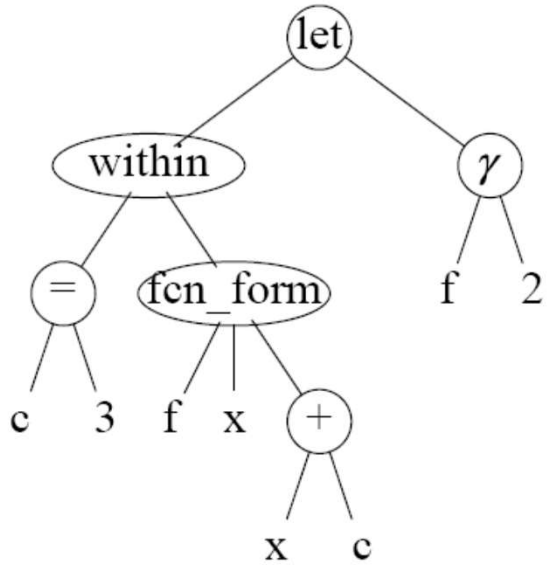
Standardizing the 'within'

- See example



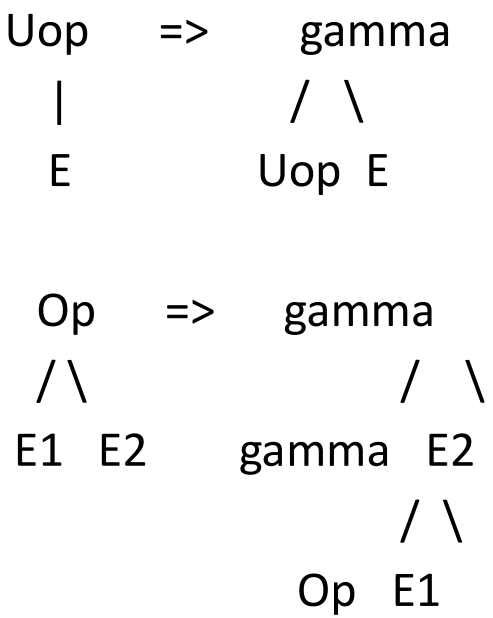
Standardizing the 'within':

```
let c = 3 within f x = x + c in f 2
```



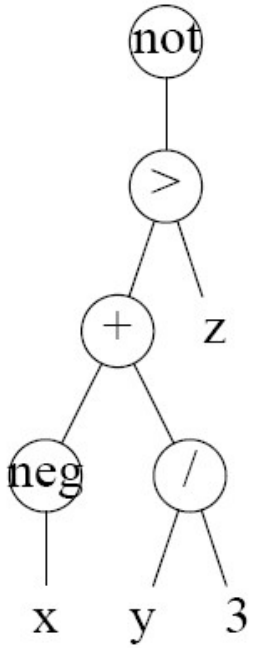
Standardizing Unary and Binary Operators

- See example



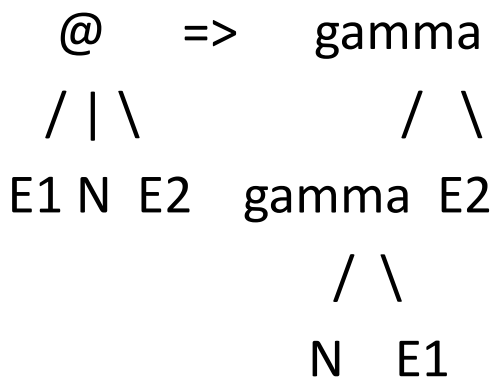
Standardizing unary and binary operators:

```
not - x + y / 3 > z
```



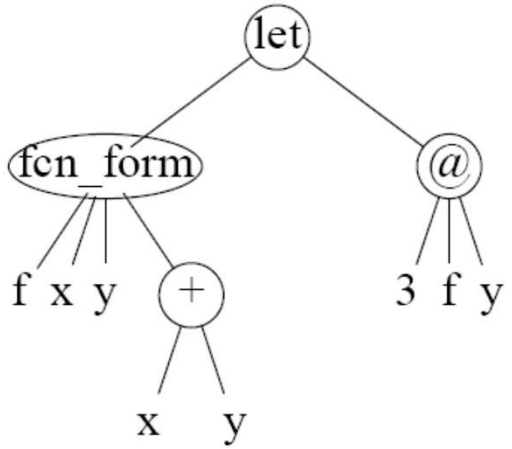
Standardizing the '@' Operator

- See example



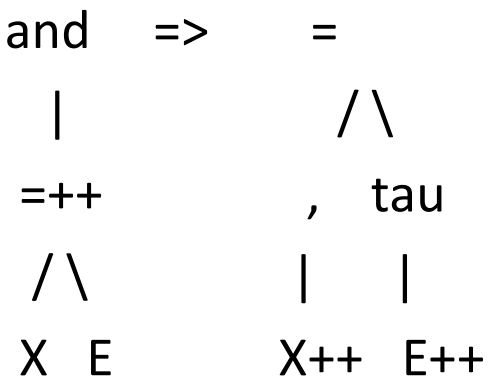
Standardizing the '@' operator:

```
let f x y = x + y in 3 @f y
```



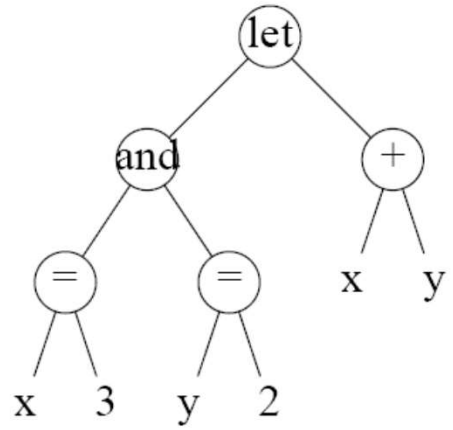
Standardizing Simultaneous Definitions

- See example

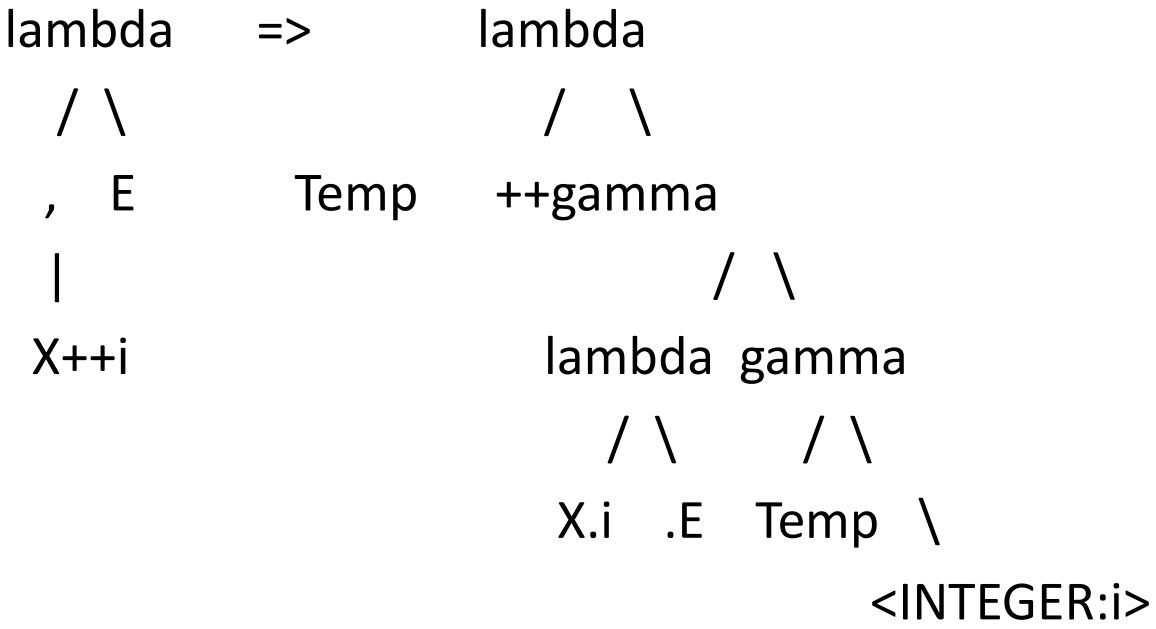


Standardizing simultaneous definitions:

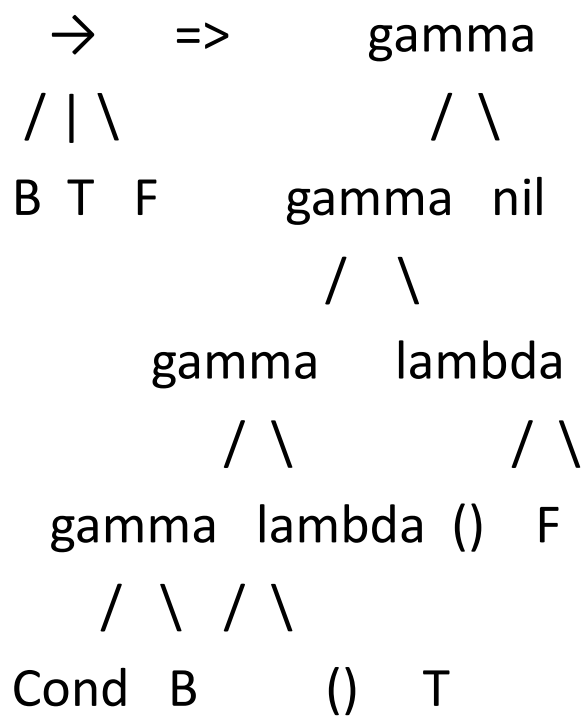
```
let x = 3 and y = 2 in x + y
```



Standardizing Simultaneous Definitions (cont'd)



Standardizing the Conditional Operator



Cond = fn B. fn T. fn F. B → T | F

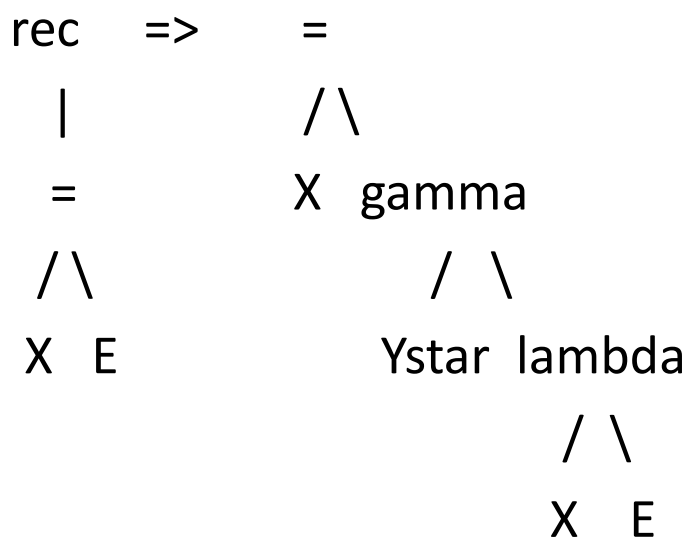
Circular semantic definition !

Circular Semantic Definitions

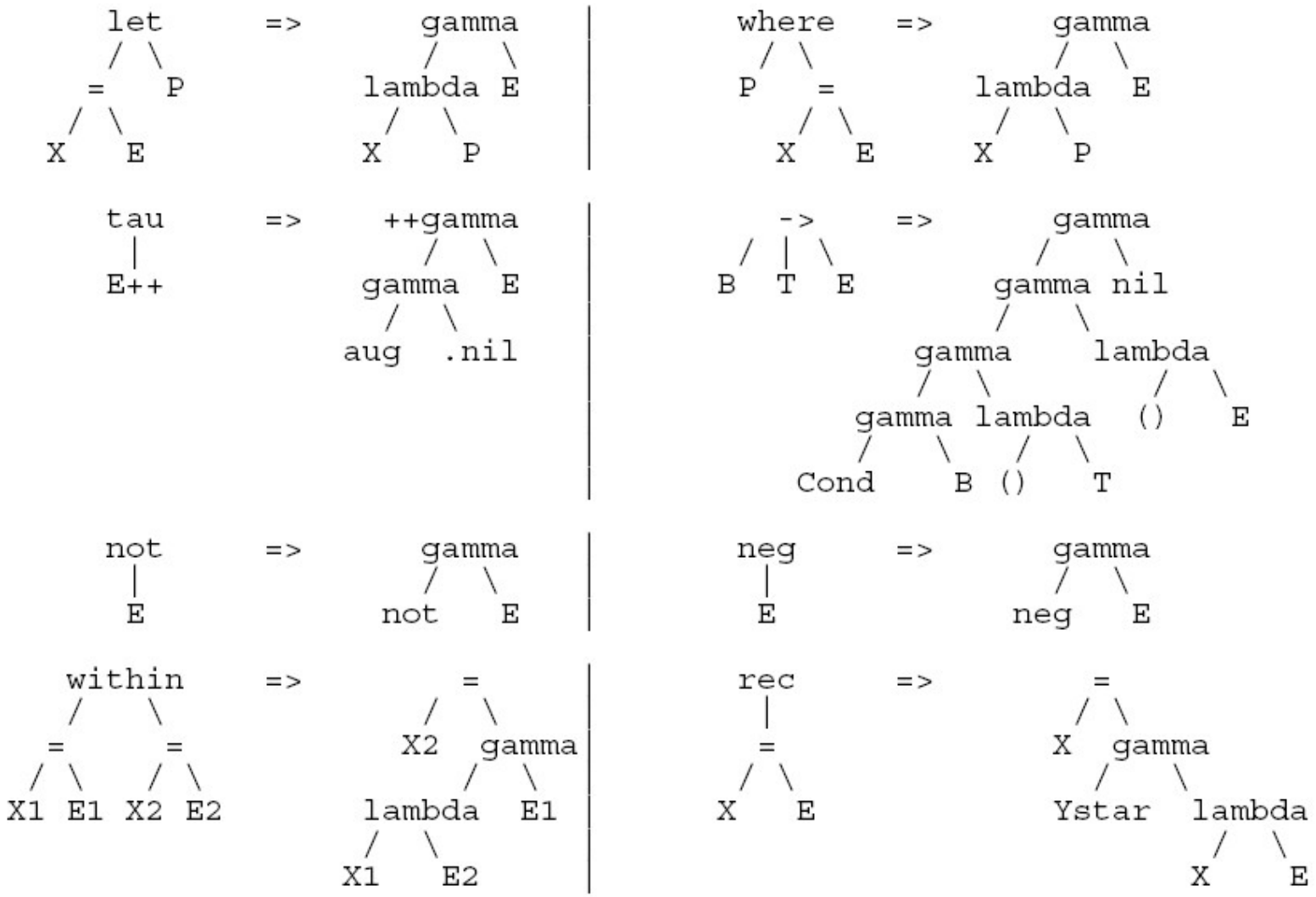
- K. Goedel's Incompleteness Theorem (1930's): Every logic system is either incomplete or inconsistent.
- Incomplete is preferable to inconsistent.
- Inevitable in semantics
- English dictionary is useless to someone who understands no English.

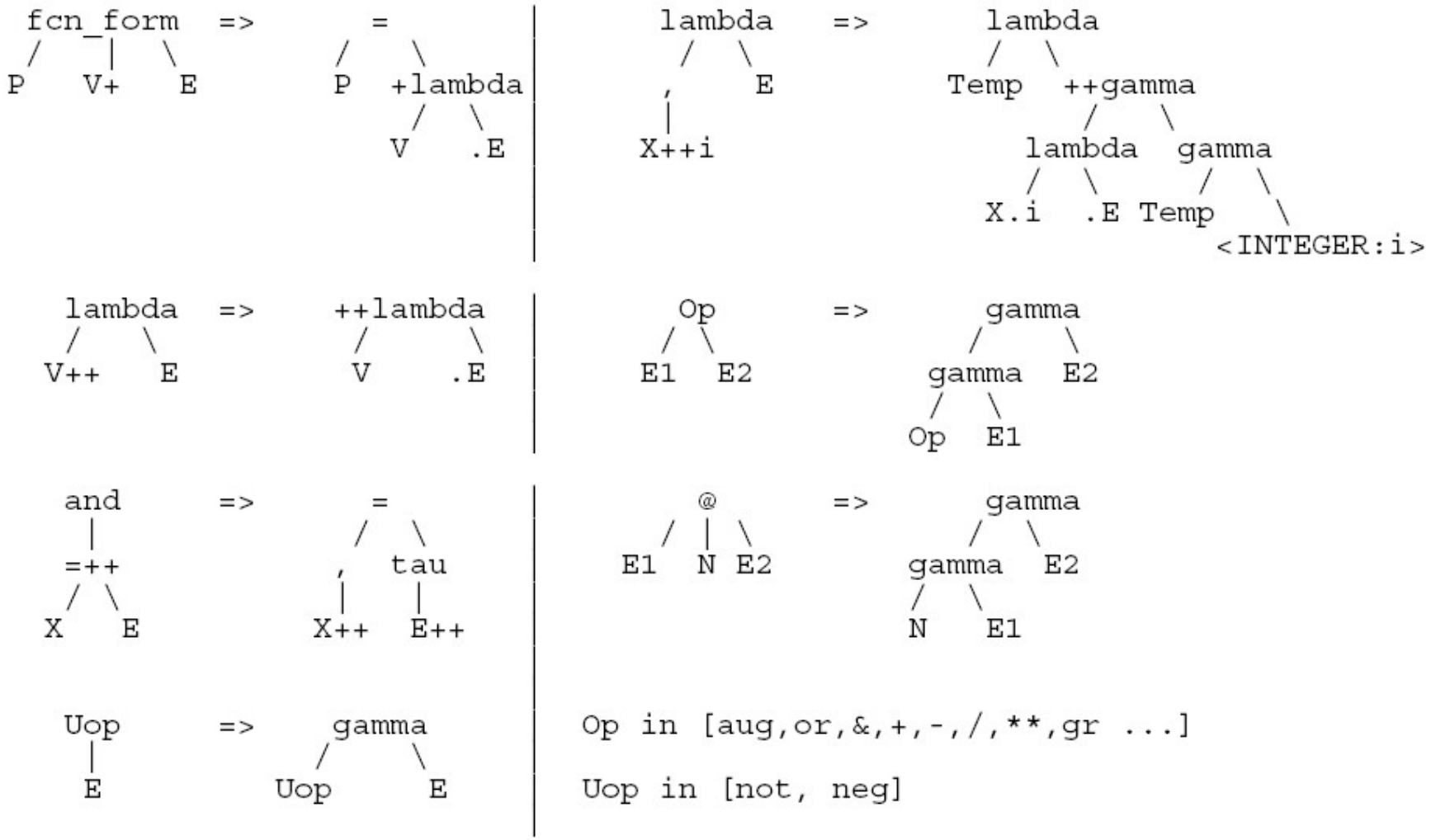
Standardizing 'rec'

- Will do later



RPAL SUBTREE TRANSFORMATIONAL GRAMMA




fcn_form => =
P V+ E P +lambda
V .E
lambda => lambda
, E Temp ++gamma
X++i lambda gamma
X.i .E Temp
<INTEGER:i>
lambda => ++lambda
V++ E V .E
Op => gamma
E1 E2 gamma E2
Op E1
and => =
=++ , tau
X E X++ E++
@ => gamma
E1 N E2 gamma E2
N E1
Uop => gamma
E Uop E
Op in [aug,or,&,+,-,/,**,gr ...]
Uop in [not, neg]

Summary

- Transform AST into a standardized tree (ST).
- ST is binary.
- All internal nodes in the ST are either gamma or lambda.
- It's called "desugaring" the program: reducing it to two constructs: function abstraction/definition (lambda), and function application (gamma).

Thank You!

REFERENCES

- Programming Language Pragmatics by Michael L. Scott. 3rd edition. Morgan Kaufmann Publishers. (April 2009).
- Lecture Slides of Dr.Malaka Walpola and Dr.Bermudez