PoPL Spring 2026

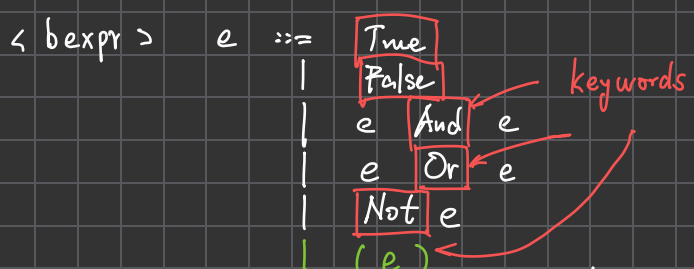
Lecture 6. Abstract & Concrete Syntax & OpSem

1. Concrete vs. Abstract Syntax

Concrete Syntax — Grammar

Backus-Naur-Form (BNF)

```
BOOL :   True
         False
         True And False
         True Or True
         Not False
```



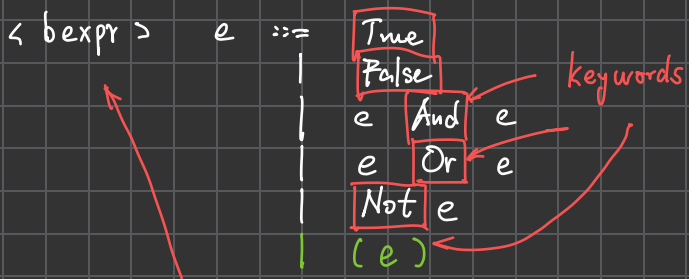
True / False : Literals / Terminals

"(True Or False) And True" ∈ bexpr ? (e) Yes

"True Not True" ∈ bexpr False

Concrete vs Abstract

Py: [1, 2, 3, ...] "*" "+" "?" "[.]"




```
type bexpr = True
           | False
           | And of bexpr * bexpr
           | Or of bexpr * bexpr
           | Not of bexpr
```

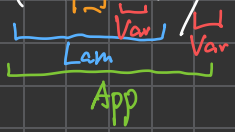
Abstract

Abstract Syntax Tree (AST)

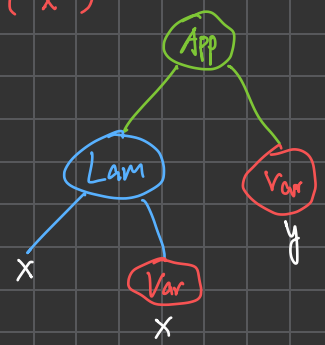
```
<λ-expr>  e ::= v          // (variable)
            | e e          // function application
            | λ v . e      //λ function def
            | ( e )
<variable>  v ::= [a-z]+
```

Concrete Syntax for λ-expression

Problem : AST for $(\lambda x.\, x)\, y$




```
App (
  Lam (
    "x", Var ("x")
  ),
  Var ("y")
)
```



2. Interpreter of a language

```
<bexpr>  e ::= True
             | False
             | e And e
             | e Or e
             | Not e
             | ( e )
```

ds

$[\![e]\!] \in \mathbb{B}$ $\quad \mathbb{B} = \{ True, False \}$

$[\![.]\!] : bexpr \to \mathbb{B}$

function : $X \to Y$

$[\![random()]\!]$

Relation = <

"1 < 2" $\Rightarrow$ true

"3 = 4" $\Rightarrow$ false

Relation of beval "$\Rightarrow$"

$\Rightarrow$: bexpr x $\mathbb{B}$ $\to$ $\mathbb{B}$

program — result — does program produce the result

"True ⇒ True" Yes

"True ⇒ False" No

Py "random() ⇒ 0.1" Yes

"random() ⇒ 3.14" No

---

in BOOL, given $p_1, p_2 \in$ bexpr, if

$p_1 = p_2$, $p_1 \Rightarrow v_1$, $p_2 \Rightarrow v_2$,

then $v_1 = v_2$

Python:

```
def loop():
    while True:
        pass
    return 1
```

loop() ⇒ 1

$\forall v.$ loop() $\not\Rightarrow v$

3. Operational Semantics

if-then $\Rightarrow : bexpr \to \mathbb{B} \to \mathbb{B}$

modus ponens $\mathbb{B} = \{T, F\}$

True $\Rightarrow$ T
False $\Rightarrow$ F

~~if $e_1 \Rightarrow T$, $e_2 \Rightarrow T$, then And$(e_1, e_2) \Rightarrow T$~~
~~if $e_1 \Rightarrow T$, $e_2 \Rightarrow F$, then And$(e_1, e_2) \Rightarrow F$~~

if $e_1 \Rightarrow v_1$, $e_2 \Rightarrow v_2$ then And$(e_1, e_2) \Rightarrow v_1 \wedge v_2$
if $e_1 \Rightarrow v_1$, $e_2 \Rightarrow v_2$ then Or$(e_1, e_2) \Rightarrow v_1 \vee v_2$

$$[\text{And}]\ \frac{e_1 \Rightarrow v_1 \qquad e_2 \Rightarrow v_2}{e_1 \text{ And } e_2 \Rightarrow v_1 \wedge v_2}$$

$$[\text{Or}]\ \frac{e_1 \Rightarrow v_1 \qquad e_2 \Rightarrow v_2}{e_1 \text{ Or } e_2 \Rightarrow v_1 \vee v_2}$$

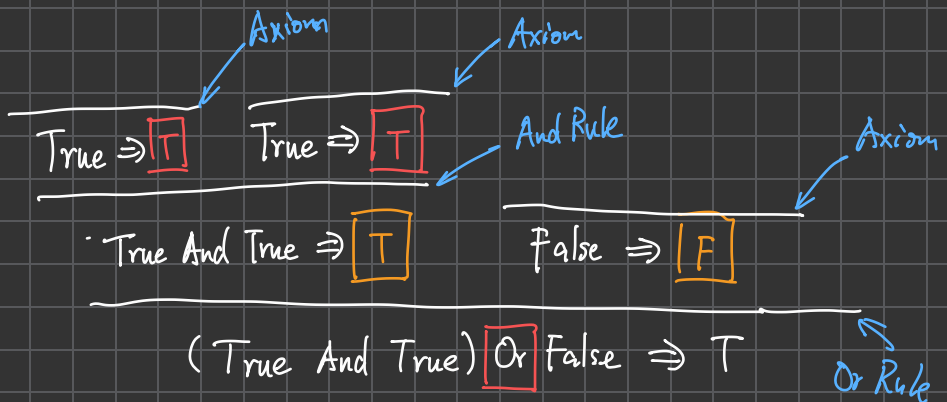
$$[\text{Not}]\ \frac{e_1 \Rightarrow v_1}{\text{Not } e_1 \Rightarrow \neg v_1}$$

$$[\text{True}]\ \frac{}{\text{True} \Rightarrow T}$$

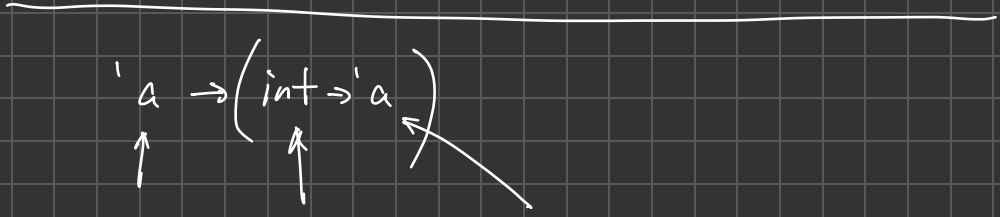
$$[\text{False}]\ \frac{}{\text{False} \Rightarrow F}$$

wedge, vee, not — Rules

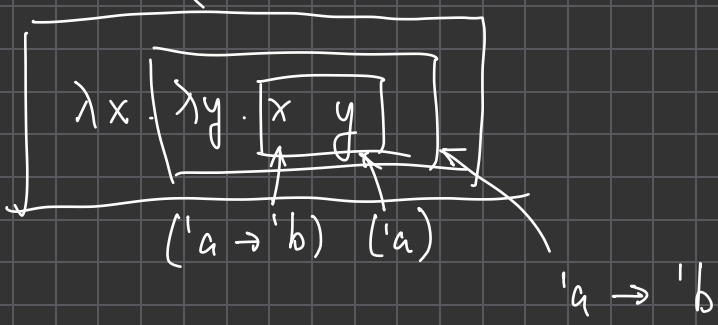
Axioms



Proof Trees



TArrow (TParam "a", TArrow ((TVar "int", (TParam "a"))))



('a → 'b) → ('a → 'b)