CSC 372, Spring 2025

SML Type Inference and Prolog Example Problems

Michelle Strout

THE UNIVERSITY OF ARIZONA

February 25, 2025

Plan

- **Announcements**
  - SA4 was due last night
  - Grades for ICA8 have been posted
  - Anki deck with prolog and type inference questions has been posted
  - LA2 will be posted tomorrow
- **Last time**
  - Some more Prolog
  - ICA8/Quiz8 about Prolog Introduction Reading assignment
  - Type inference for SML
- **Today**
  - Type inference for SML
  - Example Prolog problems

Outline for rest of today

- Type inference in SML

- Prolog example problems

SML Type Inference Rules

- Constants
  - `true, false`, type is `bool`
  - Integer literals (e.g. `42`), type is `int`
- Variable use
  - If x : tau is in the environment, then x has type tau
- Lambda functions `(fn x => e)`
  - If `x : tau1` and `e : tau2`, then `fn x => e : tau1 -> tau2`
- Function application `(e1 e2)`
  - If `e1 : tau1 -> tau2` and `e2 : tau1`, then `e1 e2 : tau2`
- Addition, or multiplication `(e1 + e2)`
  - Both `e1` and `e2` must be `int`, and the result is `int`
- If expression `(if e1 then e2 else e3)`
  - If `e1 : bool, e2 : tau, e3 : tau,` **result is** `tau`
- Let expression `(let val x = e1 in e2 end)`
  - Infer `e1 : tau1`, then infer `e2` with `x : tau1` in environment

Using an AST to help guide the process

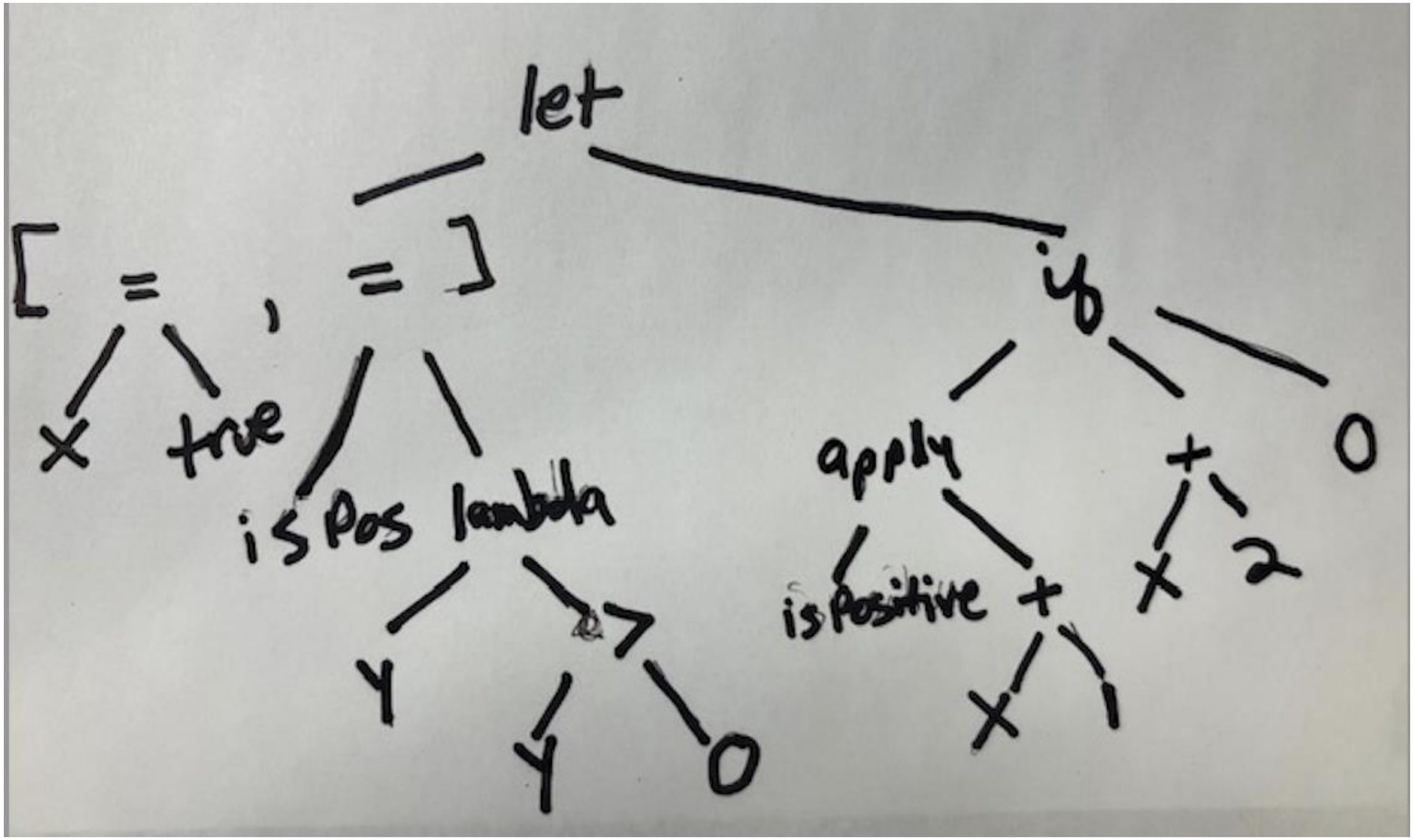
- Example

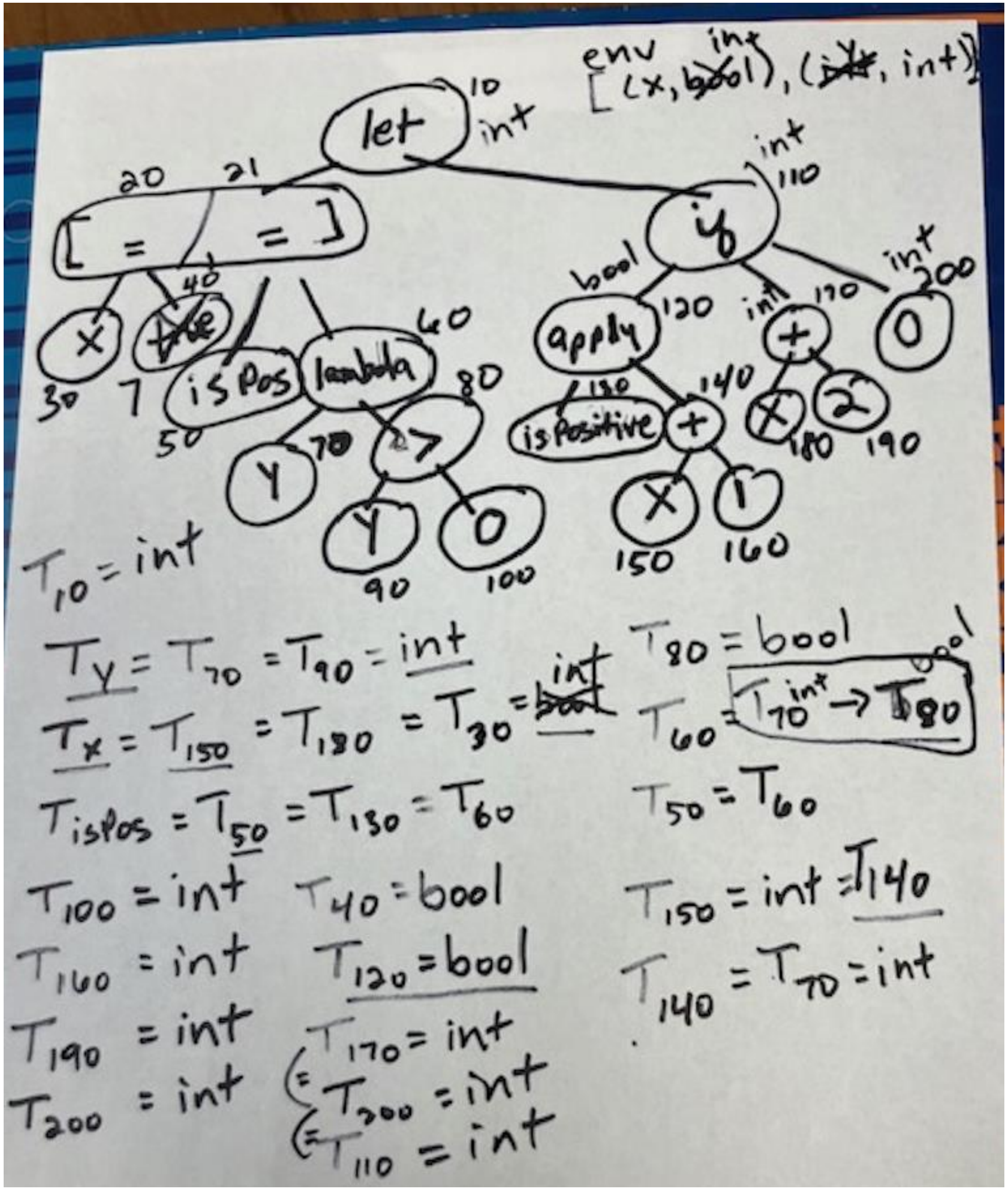
```
let
  val x = true
  val isPositive = fn y => y > 0
in
  if isPositive (x + 1) then x + 2 else 0
end
```

- Draw the AST for the example

- Apply type inference rules

AST from class





$T_{10} = int$

$T_{Y} = T_{70} = T_{90} = int$

$T_{x} = T_{150} = T_{180} = T_{30} = \cancel{bool}$ int

$T_{isPos} = T_{50} = T_{130} = T_{60}$

$T_{100} = int$

$T_{160} = int$

$T_{190} = int$

$T_{200} = int$

$T_{40} = bool$

$T_{120} = bool$

$T_{170} = int$

$= T_{200} = int$

$= T_{110} = int$

$T_{80} = bool$

$T_{60} = \boxed{\overset{int}{T_{70}} \rightarrow \overset{bool}{T_{80}}}$

$T_{50} = T_{60}$

$T_{150} = int = T_{140}$

$T_{140} = T_{70} = int$

ASTs for parts of SML

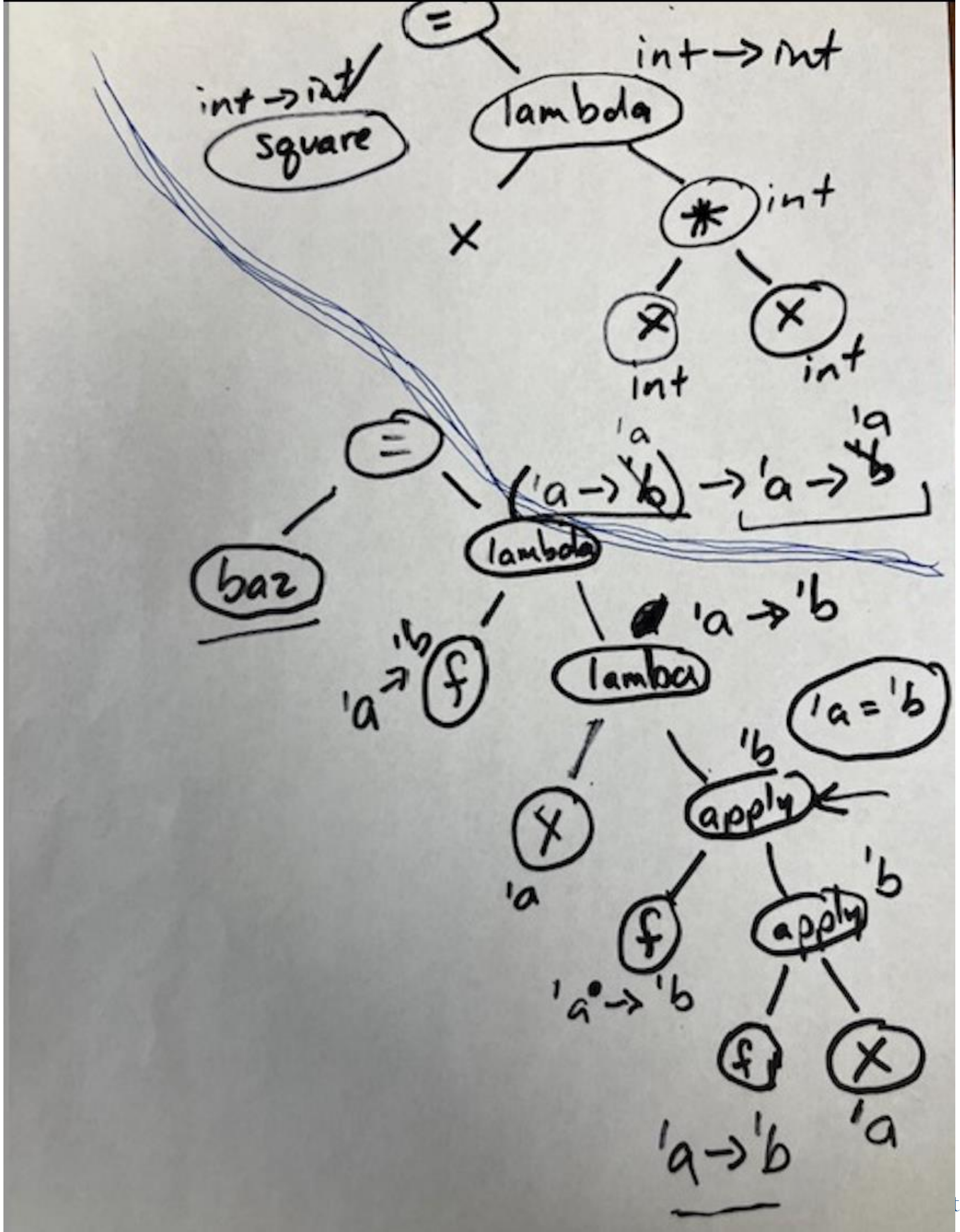
- Constants
  - `bool(true)` **and** `bool(false)`, type is `bool`
  - `int(42), int(3), int(_)`, type is `int`
- Variable use
  - `var(x),` If `(x,tau)` is in the environment, then `x` has type `tau`
- Lambda functions `(fn x => e)`
  - `lambda(x, E)`, E can be any other expression like `var(x)`
- Function application `(e1 e2)`
  - `apply(E1,E2)`
- Addition, or multiplication `(e1 + e2)`
  - `plus(E1,E2)` or `mult(E1,E2)`
- If expression `(if e1 then e2 else e3)`
  - `E=if(E1, E2, E3)`, **type of E1 must be bool, type of E2, E3, and E must all be the same (i.e., unify)**
- Let expression `(let val x = e1 in e2 end)`
  - `let(x, E1, E2),` put `(x,tau)` into the environment

AST-based approach for earlier examples

- Earlier examples

```
fun square x = x * x;
fun baz f x = f (f x);

fun pairself x = (x,x)
```


=
int->int
square
lambda
int->int
x
*
int
x
x
int
int
=
baz
lambda
('a->'b) -> 'a -> 'b
'a
'a
'a->'b
f
lambda
'a->'b
'a='b
x
'a
apply
'b
f
'a->'b
apply
'b
f
x
'a->'b
'a