CSC 372, Spring 2025

Some More Prolog and SML Type Inference

Michelle Strout

THE UNIVERSITY OF ARIZONA

February 25, 2025

Plan

- **Announcements**
  - SA4 Prolog due tomorrow/Wednesday Feb 26th
  - LA1 grades are out
- **Last time**
  - Introduction to Prolog
- **Today**
  - Some more Prolog
  - ICA8/Quiz8 about Prolog Introduction Reading assignment
  - Type inference for SML

Outline for rest of today

- Some more Prolog examples, including lists
- ICA8/Quiz8
- Type inference in SML

Exercise: Pattern matching with lists

- Try out the following queries. Write down the results.
  - X = [1,2,3,4]. % X = [1,2,3,4]
  - [X|Y] = [1,2,3,4]. % X=1, Y=[2,3,4]
  - [X,Y|Z] = [1,2,3,4]. % X=1, Y = 2, Z=[3,4]
  - [_,_,_|X] = [1,2,3,4]. % X = [4]
  - [A,B,C,D|E] = [1,2,3,4]. % A=1, B=2, C=3, D=4, E=[]

- Explain what each of the following mean: [], |, _
  - | is acting like :: in SML, mostly
  - A::B::C::D::E
  - _ same as SML in that nothing is bound

Exercise

- Predict the results of the following queries.
  - [A,B] = [1,2].
  - [A,B,C] = [1,2].
  - [A,B|C] = [1,2].
  - [_|A] = [1,2].

Tracing through the 372-grade.pl

THE UNIVERSITY OF ARIZONA
Computer Science

- **Handy prolog commands**
  - ["372-grade"]. % loads the prolog database(DB) of facts and rules
  - make. % reloads
  - listing % shows all facts and rules available in DB
  - assert(<fact or rule>). % make fact or rule available in DB
  - trace. % starts a trace
  - notrace. % leave trace mode
  - When doing a trace
    - Hit enter to step/creep.
    - Hit 'a' to abort the current execution.

- **Referencing variables from previous queries**

```
?- listing
?- assert(foo(A,B):- A=B).
?- assert(baz(X) :- 700=X).
?- MyVar = 200, foo(300, AnotherVar).
?- baz($MyVar), X is $AnotherVar + 2.
```

ICA8: Quiz on Prolog Intro

- Read the instructions on the quiz

Type Inference in Standard ML (SML)

- **Hindley-Milner Type System**
  - Strongly typed, statically checked without explicit type annotations
  - Supports parametric polymorphism
- **Type Inference via Algorithm W (Damas-Milner 1982)**
  - Automatically deduces types using unification
  - Works by recursively analyzing expressions and constraints
- **Unification and Type Constraints**
  - Type variables are assigned and unified to resolve constraints
  - Merges equivalent types while detecting inconsistencies

Example from reading assignment

THE UNIVERSITY OF ARIZONA
Computer Science

- What is the type of the following SML function and why?

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

- The type of pairself is

```
fn: ‘a -> ‘a * ‘a
```

- Why?
  - Know return type is a tuple because of tuple (x,x) on rhs
  - x is a parameter, we don’t know type, so using type variable ‘a
  - Since returning (x,x) tuple, those var accesses have same type
  - In a function definition, so have ‘a -> ‘a * ‘a

Example from MT1

- What is the type of the following SML function and why?

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

- The type of square is

```
int -> int
```

- Why?
  – Because the multiplication operator ‘*’ defaults to int -> int -> int or (int * int) -> int

Example from MT1

THE UNIVERSITY OF ARIZONA
Computer Science

- What is the type of the following SML function and why?

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

- The type of baz is

- Why?
  - f as ‘a -> ‘a
  - (‘a -> ‘a) -> a’ -> a’

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

ASTs for parts of SML

THE UNIVERSITY OF ARIZONA. Computer Science

- 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

The University of Arizona
Computer Science

- Earlier examples

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

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