



CSCI-344

Programming Language Concepts (Section 3)

Lecture 24

Type Inference

Instructor: Hossein Hojjat

November 7, 2016

Done:

- Static Typing
- Typed Impcore: Monomorphic type system
- Typed μ Scheme: Polymorphic type system

This session:

- Type Inference

- Nano-ML: type-safe bridge language which is derived from μ Scheme
- Programmer never writes explicit type annotations
- Nano-ML uses type inference
- Type of every parameter, variable, and function is inferred

Nano-ML vs. μ Scheme

- Different type systems
- No mutation in nano-ML
 - No assignment to variables, no override on existing binding
- `val-rec` in nano-ML allows recursive calls to itself

Type Inference Example

μ Scheme

```
-> (define double (x) (+ x x))  
double
```

Typed μ Scheme

```
-> (define int double ([x : int]) (+ x x))  
double : (int -> int)
```

Nano-ML

```
-> (define double (x) (+ x x))  
double : (int -> int)
```

Type Inference Example

The compiler infers types: there is lower annotation burden

Typed μ Scheme

```
-> ((@ cons int) 1 ((@ cons int) 1 (@ '() int)))  
(1 1) : (list int)
```

Nano-ML

```
-> (cons 1 (cons 1 '()))  
(1 1) : (list int)
```

Type Inference Problem

- **Input:** term with no type annotations
- **Output:** valid typing for term, or, show that it has no valid typing

Key Steps

1. Assign a preliminary type to every sub-expression
 - Use fresh type variables to represent unknown types
2. Generate equality constraints among the types and type variables
3. Solve the constraints to compute a typing

Undecidability

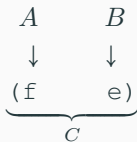
- Undecidable for a language as powerful as Typed μ Scheme

Generate Constraints

Constraint generation for function application and definition

Application

- If type of f is A , type of e is B , type of $(f\ e)$ is C , then
 $A = B \rightarrow C$



$$A = B \rightarrow C$$

Definition

- If type of f is A , type of x is B , type of e is C , then $A = B \rightarrow C$



$$A = B \rightarrow C$$

Example 1

```
-> (val-rec double (lambda (x) (+ x x)))  
double : (int -> int)
```


Example 1

```
-> (val-rec double (lambda (x) (+ x x)))  
double : (int -> int)
```

Step 1: Assign preliminary types

Sub-expression	Preliminary Type
double	A
x	B
(+ x x)	C
+	$\text{int} \rightarrow \text{int} \rightarrow \text{int}$

Example 1

```
-> (val-rec double (lambda (x) (+ x x)))  
double : (int -> int)
```

Step 1: Assign preliminary types

Sub-expression	Preliminary Type
double	A
x	B
$(+ x x)$	C
$+$	$\text{int} \rightarrow \text{int} \rightarrow \text{int}$

Step 2: Generate constraints

$$\begin{aligned}\text{int} \rightarrow \text{int} \rightarrow \text{int} &= B \rightarrow B \rightarrow C \\ A &= B \rightarrow C\end{aligned}$$

Example 1

```
-> (val-rec double (lambda (x) (+ x x)))  
double : (int -> int)
```

Step 1: Assign preliminary types

Sub-expression	Preliminary Type
double	A
x	B
$(+ x x)$	C
$+$	$\text{int} \rightarrow \text{int} \rightarrow \text{int}$

Step 2: Generate constraints

$$\begin{aligned}\text{int} \rightarrow \text{int} \rightarrow \text{int} &= B \rightarrow B \rightarrow C \\ A &= B \rightarrow C\end{aligned}$$

Step 3: Solve constraints

$$A = \text{int} \rightarrow \text{int}, B = \text{int}, C = \text{int}$$

Example 2

```
-> (val-rec f (lambda (x) (if x 1 0)))  
?
```

Example 2

```
-> (val-rec f (lambda (x) (if x 1 0)))  
f : (bool -> int)
```

Step 1: Assign preliminary types

Sub-expression	Preliminary Type
<code>f</code>	A
<code>x</code>	B
<code>(if x 1 0)</code>	C
<code>if</code>	$\forall\alpha. \text{bool} \rightarrow \alpha \rightarrow \alpha \rightarrow \alpha$
<code>0, 1</code>	int

Step 2: Generate constraints

$$\begin{aligned} \text{bool} \rightarrow \alpha_1 \rightarrow \alpha_1 \rightarrow \alpha_1 &= B \rightarrow \text{int} \rightarrow \text{int} \rightarrow C \\ A &= B \rightarrow C \end{aligned}$$

Step 3: Solve constraints

$$A = \text{bool} \rightarrow \text{int}, \quad B = \text{bool}, \quad C = \alpha_1 = \text{int}$$

Examples on the Board

```
-> (val f (lambda (x) (if x x (+ 1 x))))  
?
```

Examples on the Board

```
-> (val f (lambda (x) (if x x (+ 1 x))))  
type error: cannot make int equal to bool
```

Examples on the Board

```
-> (val f (lambda (x) (if x x (+ 1 x))))  
type error: cannot make int equal to bool
```

```
-> (val f (lambda (x f) (f x)))  
?
```


Examples on the Board

```
-> (val f (lambda (x) (if x x (+ 1 x))))  
type error: cannot make int equal to bool
```

```
-> (val f (lambda (x f) (f x)))  
f : (forall ('a 'b) ('a -> ('a -> 'b) -> 'b))
```

Examples on the Board

```
-> (val f (lambda (x) (if x x (+ 1 x))))  
type error: cannot make int equal to bool
```

```
-> (val f (lambda (x f) (f x)))  
f : (forall ('a 'b) ('a -> ('a -> 'b) -> 'b))
```

```
-> (val f (lambda (xs) (car (car xs))))  
?
```

Examples on the Board

```
-> (val f (lambda (x) (if x x (+ 1 x))))  
type error: cannot make int equal to bool
```

```
-> (val f (lambda (x f) (f x)))  
f : (forall ('a 'b) ('a -> ('a -> 'b) -> 'b))
```

```
-> (val f (lambda (xs) (car (car xs))))  
f: (forall 'a . 'a list list -> 'a)
```

This Lecture

- Type Inference: Example

Next Lecture

- Formalizing Type Inference