FP: Oct. 17

Ken Wakita

MinCaml types

Type (τ)	Explanation	Examples of members
π	Primitive type	true/false: bool 0, ±1, ±2, ±3, ….: for int 3.14: float "string": string
$\tau 1 \rightarrow \ldots \rightarrow \tau n \rightarrow \tau$	Functional type	# cos;; - : float -> float = <fun>
$\tau 1 \times \ldots \times \tau n$	Tupple type	(true, 1, 3.14);; - : bool * int * float = (true, 1, 3.14)
τ array	Array type	# Array.create 3 1.0;; - : float array = [| 1.; 1.; 1. |]
α	Type variable (introduced for the	(whatever)

x^a

Tour guide: Typing rule ABC

- const
- operators
- variable usage and definition
- tuple and array
- array indexing
- conditional
- pattern matching against tuples
- function application
- let rec

op is a primitive operator that takes values of π1, …, πn and gives a value of π

c is a constant member of π

$$\frac{c \text{ は } \pi \text{ 型の定数}}{\Gamma \vdash c : \pi}$$

1

$$\frac{\Gamma \vdash e_1 : \pi_1 \quad \ldots \quad \Gamma \vdash e_n : \pi_n \quad op \text{ は } \pi_1, \ldots, \pi_n \text{ 型の値を受け取って } \pi \text{ 型の値を返すプリミティブ演算}}{\Gamma \vdash op(e_1, \ldots, e_n) : \pi}$$

2

$$\frac{\Gamma \vdash e_1 : \texttt{bool} \quad \Gamma \vdash e_2 : \tau \quad \Gamma \vdash e_3 : \tau}{\Gamma \vdash \texttt{if } e_1 \texttt{ then } e_2 \texttt{ else } e_3 : \tau}$$

A

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash \texttt{let } x = e_1 \texttt{ in } e_2 : \tau_2}$$

4

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x : \tau}$$

3

$$\frac{\Gamma, x : \tau_1 \to \ldots \to \tau_n \to \tau, y_1 : \tau_1, \ldots, y_n : \tau_n \vdash e_1 : \tau \qquad \Gamma, x : \tau_1 \to \ldots \to \tau_n \to \tau \vdash e_2 : \tau'}{\Gamma \vdash \texttt{let rec } x\ y_1\ \ldots\ y_n = e_1 \texttt{ in } e_2 : \tau'}$$

C

$$\frac{\Gamma \vdash e : \tau_1 \to \ldots \to \tau_n \to \tau \qquad \Gamma \vdash e_1 : \tau_1 \quad \ldots \quad \Gamma \vdash e_n : \tau_n}{\Gamma \vdash e\ e_1\ \ldots\ e_n : \tau}$$

B

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad \ldots \quad \Gamma \vdash e_n : \tau_n}{\Gamma \vdash (e_1, \ldots, e_n) : \tau_1 \times \ldots \times \tau_n}$$

5

$$\frac{\Gamma \vdash e_1 : \tau_1 \times \ldots \times \tau_n \quad \Gamma, x_1 : \tau_1, \ldots, x_n : \tau_n \vdash e_2 : \tau}{\Gamma \vdash \texttt{let } (x_1, \ldots, x_n) = e_1 \texttt{ in } e_2 : \tau}$$

8

$$\frac{\Gamma \vdash e_1 : \texttt{int} \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \texttt{Array.create } e_1\ e_2 : \tau \texttt{ array}}$$

6

$$\frac{\Gamma \vdash e_1 : \tau \texttt{ array} \quad \Gamma \vdash e_2 : \texttt{int}}{\Gamma \vdash e_1.(e_2) : \tau}$$

7

$$\frac{\Gamma \vdash e_1 : \tau \texttt{ array} \quad \Gamma \vdash e_2 : \texttt{int} \quad \Gamma \vdash e_3 : \tau}{\Gamma \vdash e_1.(e_2) \leftarrow e_3 : \texttt{unit}}$$

9

図 3: MinCaml の型つけ規則

MinCaml's typing rule

Typing: preliminary

- Typing environment (Γ): a set of typing assumption (e.g.: $\Gamma = x$: σ means "x has type σ under Γ)

- Typing relation (Γ |- e: σ): e is an expression of type σ under Γ

- Typing rule: Above the line: the premise / Below the line: conclusion

Interpretation of the typing rule for function definition

$$\frac{\Gamma, x : \tau_1 \rightarrow \ldots \rightarrow \tau_n \rightarrow \tau, y_1 : \tau_1, \ldots, y_n : \tau_n \vdash e_1 : \tau \qquad \Gamma, x : \tau_1 \rightarrow \ldots \rightarrow \tau_n \rightarrow \tau \vdash e_2 : \tau'}{\Gamma \vdash \texttt{let rec } x\ y_1\ \ldots\ y_n = e_1 \texttt{ in } e_2 : \tau'}$$

- Assumption
  - Γ, *fadd* : **float** → **float** → **float**, *y1* : **float**, *y2* : **float** |- *y1* +. *y2* : **float**
  - Γ, *fadd* : **float** → **float** → **float** |- *fadd* 3.0 (5.0 +. 7.0) : **float**
    (from function application)
- Conclusion
  Γ |- **let rec** *fadd y1 y2* = *y1* +. *y2* **in** *fadd* 3.0 (5.0 +. 7.0) : **float**

Interpretation of functional application

$$\frac{\Gamma \vdash e : \tau_1 \rightarrow \ldots \rightarrow \tau_n \rightarrow \tau \qquad \Gamma \vdash e_1 : \tau_1 \quad \ldots \quad \Gamma \vdash e_n : \tau_n}{\Gamma \vdash e\ e_1\ \ldots\ e_n : \tau}$$

- Assumption
  - Γ |- *fadd* : **float → float → float**
  - Γ |- 3.0 : **float** Γ |- (5.0 +. 7.0) : **float**
- Conclusion
  Γ |- *fadd* 3.0 (5.0 +. 7.0) : **float**