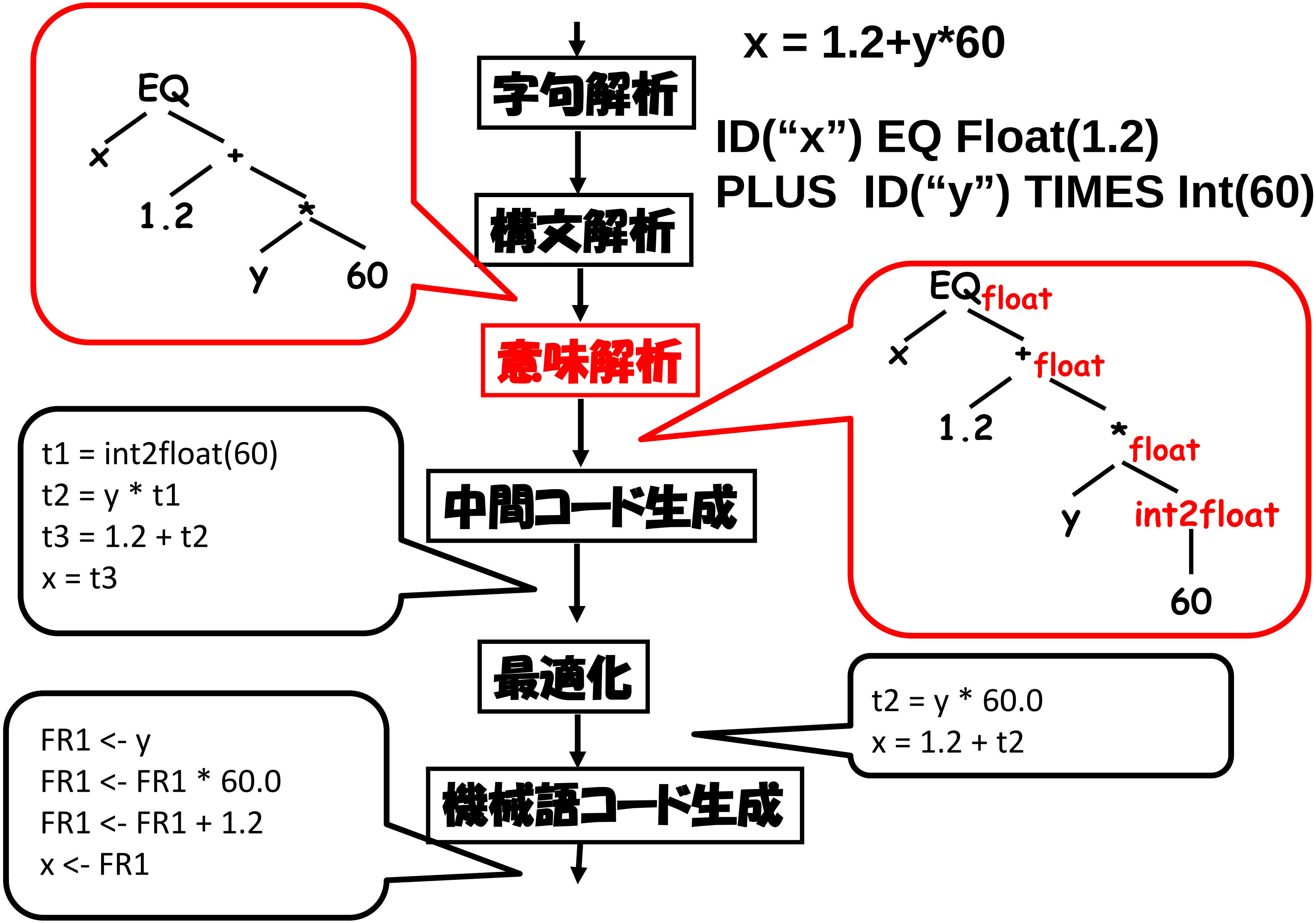
意味解析

コンパイラの構造


x = 1.2+y*60
字句解析
ID("x") EQ Float(1.2)
PLUS ID("y") TIMES Int(60)
構文解析
EQ
x
+
1.2
*
y
60
意味解析
EQ float
x
+ float
1.2
* float
y
int2float
60
中間コード生成
t1 = int2float(60)
t2 = y * t1
t3 = 1.2 + t2
x = t3
最適化
t2 = y * 60.0
x = 1.2 + t2
機械語コード生成
FR1 <- y
FR1 <- FR1 * 60.0
FR1 <- FR1 + 1.2
x <- FR1

意味解析

- 入力: 抽象構文木
- 出力: 付加情報つき抽象構文木

- 役割:
  - 構文だけでは判別がつかない誤りの検出
    - 型エラー (e.g. var x:int=1; print_string x;)
    - 変数宣言の誤り (e.g. var x1:int = 1; x := x+2;)
  - 後の処理に必要な情報を収集
    - 変数の参照関係
    - 演算子のオーバーローディングの解消(=, < など)

変数の参照関係

```
function f(v:int) =
let var v := v+1
in print(v);
   let var v:= v+1 in
   print(v) end;
   print(v)
end
```

変数の参照関係

```
function f(v:int) =
let var v := v+1
in print(v);
  let var v:= v+1 in
  print(v) end;
  print(v)
end
```

変数の参照関係

```
function f(v:int) =
let var v := v+1
in print(v);
  let var v:= v+1 in
  print(v) end;
  print(v)
end
```

変数の参照関係

```
function f(v:int) =
let var v := v+1
in print(v);
  let var v:= v+1 in
  print(v) end;
  print(v)
end
```

変数の参照関係

```
function f(v:int) =
let var v := v+1
in print(v);
  let var v:= v+1 in
  print(v) end;
  print(v)
end
```

変数の参照関係

```
function f(v:int) =
let var v := v+1
in print(v);
   let var v:= v+1 in
   print(v) end;
   print(v)
end
```

→

```
function f(v1:int) =
let var v2 := v1+1
in print(v2);
   let var v3:= v2+1 in
   print(v3) end;
   print(v2)
end
```

意味解析

- 入力：　抽象構文木
- 出力：　付加情報つき抽象構文木

- 役割：
  - 構文だけでは判別がつかない誤りの検出
    - 型エラー　(e.g.　var x:int=1; print_string x;)
    - 変数宣言の誤り　(e.g. var x1:int = 1; x := x+2;)
  - 後の処理に必要な情報を収集
    - 変数の参照関係
    - 演算子のオーバーローディングの解消(=, < など)

アウトライン

- 型検査
  - 型システム
  - 型検査アルゴリズム
- 変数の参照関係等の明確化

型検査

- 各式、変数の値を推論し、型の不整合がないかを検査
  - 変数の型宣言と代入される型の整合性
    - ✘ var x: int = 1; x = “abc”;
  - 関数または演算子が要求する型と実引数の型の整合性
    - ×var x: string=“abc”; y = x+1
- 型情報を用いて演算子のオーバーローディング等を解消
  - var x:int =1; x+2 → var x:int=1; $x+_{int}$ 2

型検査によるプログラミング言語の分類

- 静的に型付けされた(statically-typed)言語
  - コンパイル時に型検査: ML, C, Java など
- 動的に型付けされた(dynamically-typed)言語
  - 実行時に型検査: Scheme, Javascriptなど
- 強く型付けされた(strongly-typed)言語
  - 型検査を通過した後のプログラムは実行時に型の不整合をおこさない: ML, Haskell など
- 弱く型付けされた(weakly-typed)言語
  - 型検査を通過した後も型の不整合が起きる可能性あり: Cなど
    (e.g. “ int x = 1; int *p = (int *) x; *p+2;”)

型システム

(cf. 正規表現 for 字句解析、文脈自由文法 for 構文解析)

- プログラム中の各式がどのような型を持つかを推論規則(typing rules, 型付け規則)の形で表現したもの

e.g.
$$\frac{\vdash e_1 : \mathrm{int} \qquad \vdash e_2 : \mathrm{int}}{\vdash e_1 + e_2 : \mathrm{int}}$$

「 e_1 がint型であり、e_2がint型であればe_1+e_2も int型を持つ」

- 型システムから(多くの場合)型検査アルゴリズムを自明に(*)構成可能
  - tcheck(e) =

```
match e with
  e1+e2  -> if tcheck(e1)=tcheck(e2)=int then int
 | …
```

(*) ただし型検査が自明でないどころか決定不能なものもある(e.g. $F_{<}$)

型判断(type judgment)

- 与えられた式の型は文脈に依存
  - ✓ var x:int := 1; ……　　x+1　….
  - ✗ var x:string:="ab"; ……　x+1　…
- 型判断: $x_1 : \tau_1, \ldots, x_n : \tau_n \mid\!- \ e : \tau$

  「変数x_1, …, x_nがそれぞれ型$\tau_1, \ldots, \tau_n$を持つという仮定のもとで、式e は型τを持つ」

  (「実行時に変数$x_1, \ldots, x_n$にそれぞれ型$\tau_1, \ldots, \tau_n$の値が格納されていれば、eを安全に評価でき、評価結果は型τの値である」)

  - ✓　x:int |- x+1:int
  - ✗　x:string |- x+1:int
- $x_1 : \tau_1, \ldots, x_n : \tau_n$ の部分を**型環境**と呼ぶ

型付け規則

- 型判断の導出規則

e.g.

$$\frac{\Gamma \vdash e_1:\ \mathrm{int} \qquad\qquad \Gamma \vdash e_2:\ \mathrm{int}}{\Gamma \vdash e_1+e_2:\ \mathrm{int}}$$

型環境

mini-Tigerの構文

e (式) ::= n (整数)
| s (文字列)
| x (変数)
| x := e
| e op e (op $\in$ {+, *, =, <, <=})
| e_1; e_2
| if e_1 then e_2 [else e_3]
| f(e_1,...,e_n)
| let d in e

d (宣言) ::= var x:τ = e
| fun f (x_1:τ_1,...,x_n:τ_n): τ = e

τ (型) ::= b | (b_1,..., b_n)$\rightarrow$b

b (基本型) ::= unit | int | string

mini-Tiger用の型判断

- 式の型判断

  Γ |- e: τ

  「各変数xが型Γ(x)を持つ環境のもとでeを評価すると、結果の型はτになる」

  e.g. x:int, f: (int)->int |- f(x): int

- 宣言の型判断

  Γ |- d : Δ

  「各変数xが型Γ(x)を持つ環境のもとで、宣言dはΔに従った環境を生成する」

  e.g. x:int |- (var y:int := x+1) : (y:int)

型付け規則(変数)

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

ただし

$(\Gamma, x{:}\tau)(x) = \tau$

$(\Gamma, y{:}\tau)(x) = \Gamma(x) \quad (\text{if } x \neq y)$

$\varepsilon(x) =$ 未定義

e.g. (x:string, x:int)(x) = int

型付け規則(変数、定数)

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

$$\frac{}{\Gamma \vdash n : \mathrm{int}}$$

$$\frac{}{\Gamma \vdash s : \mathrm{string}}$$

$$\frac{\Gamma \vdash e1 : b1 \quad \Gamma \vdash e2 : b2 \quad (b1, b2) \to b \in \mathrm{CType}(op)}{\Gamma \vdash e1\ op\ e2 : b}$$

型付け規則(代入、制御)

$$\frac{\Gamma(x) = \tau \qquad \Gamma \vdash e : \tau}{\Gamma \vdash x := e : \mathrm{unit}}$$

$$\frac{\Gamma \vdash e_1 : \mathrm{unit} \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1 \,;\, e_2 : \tau}$$

$$\frac{\Gamma \vdash e_0 : \mathrm{int} \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \mathrm{if}\ e_0\ \mathrm{then}\ e_1\ \mathrm{else}\ e_2 : \tau}$$

型付け規則（関数呼び出し、let文）

$$\frac{\Gamma \vdash f: (\tau_1, \ldots, \tau_n) \to \tau \quad \Gamma \vdash e_i: \tau_i \quad (\text{for } i=1, \ldots, n)}{\Gamma \vdash f(e_1, \ldots, e_n): \tau}$$

$$\frac{\Gamma \vdash d : \Delta \qquad \Gamma, \Delta \vdash e: \tau}{\Gamma \vdash \text{let } d \text{ in } e: \tau}$$

型付け規則(宣言)

$$\frac{\Gamma \vdash e : b}{\Gamma \vdash \mathbf{var}\ x{:}b := e \ : (x{:}b)}$$

$$\frac{\Gamma,\ f{:}(b_1,\ldots,b_n)\to b\ ,\ x_1{:}b_1,\ldots,x_n{:}b_n \ \vdash e : b}{\Gamma \vdash \mathbf{fun}\ f(x_1{:}b_1,\ldots,x_n{:}b_n){:}b := e \ : (f{:}(b_1,\ldots,b_n)\to b)}$$

型判断の導出例

<黒板で>

```
|-
 let var x:int := 1 in
 let fun f(y:int):int := y+x in
   f (x)
 : int
```

アウトライン

- 型検査
  - 型システム
  - 型検査アルゴリズム
- 変数の参照関係等の明確化

型検査アルゴリズム

tc(Γ,e): $\Gamma \vdash$ e: τ を満たすτ を返す
（なければエラー）

型付け規則を下から読めばよい。

tc(Γ, x) = if Γ(x) is defined then Γ(x)
else error("x is undefined")

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

型検査アルゴリズム

tc(Γ,e): $\Gamma \vdash e:\ \tau$ を満たすτ を返す

（なければエラー）

型付け規則を下から読めばよい。

tc(Γ, x) = if Γ(x) is defined then Γ(x)
else error("x is undefined")

tc(Γ, n) = int

tc(Γ, s) = string

$$\frac{}{\Gamma \vdash n:\ \mathit{int}}$$

$$\frac{}{\Gamma \vdash s:\ \mathit{string}}$$

型検査アルゴリズム

$tc(\Gamma, x)$ = if $\Gamma(x)$ is defined then $\Gamma(x)$
else error("x is undefined")

$tc(\Gamma, n)$ = int

$tc(\Gamma, s)$ = string

$tc(\Gamma, e1\ op\ e2)$ =
let b1 = $tc(\Gamma, e1)$ in let b2 = $tc(\Gamma, e2)$ in
if $(b1,b2) \to b \in CType(op)$ for some b
then b else error("type mismatch on operator op")

$$\frac{\Gamma \vdash e1:b1 \quad \Gamma \vdash e2:b2 \quad (b1,b2)\to b \in CType(op)}{\Gamma \vdash e1\ op\ e2 : b}$$

型検査アルゴリズム

tc(Γ, x) = if Γ(x) is defined then Γ(x)
else error("x is undefined")

tc(Γ, n) = int

tc(Γ, s) = string

tc(Γ, e1 op e2) =
let b1 = tc(Γ, e1) in let b2 = tc(Γ, e2) in
if (b1,b2)$\rightarrow$b $\in$ CType(op) for some b
then b else error("type mismatch on operator op")

tc(Γ, x := e) = let τ1= Γ(x) in let τ2= tc(Γ, e) in
if τ1= τ2 then unit else error(...)

$$\frac{\Gamma(x) = \tau \qquad \Gamma \vdash e : \tau}{\Gamma \vdash x := e : \mathrm{unit}}$$

型検査アルゴリズム

…

$\mathrm{tc}(\Gamma, e_1\ ;\ e_2) = \mathrm{let}\ \tau 1 = \mathrm{tc}(\Gamma, e_1)\ \mathrm{in}\ \mathrm{let}\ \tau = \mathrm{tc}(\Gamma, e_2)\ \mathrm{in}$

$\mathrm{if}\ \tau 1 = \mathrm{unit}\ \mathrm{then}\ \tau\ \mathrm{else}\ \mathrm{error}(\ldots)$

$$\frac{\Gamma \vdash e_1 : \mathrm{unit} \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1\ ;\ e_2 : \tau}$$

型検査アルゴリズム

...

$tc(\Gamma, e_1 \ ; \ e_2)$ = let $\tau 1$= $tc(\Gamma, e_1)$ in let τ= $tc(\Gamma, e_2)$ in
if $\tau 1$= unit then τ else error(...)

$tc(\Gamma$, if e_0 then e_1 else $e_2)$ =
let τ_i= $tc(\Gamma, e_i)$ (for i=0,1,2) in
if τ_0= int and $\tau_1 = \tau_2$ then τ_1 else error(...)

$$\frac{\Gamma \vdash e_0 : \mathrm{int} \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \mathrm{if}\ e_0\ \mathrm{then}\ e_1\ \mathrm{else}\ e_2 : \tau}$$

型検査アルゴリズム

...

$tc(\Gamma, e_1 ; e_2)$ = let $\tau 1 = tc(\Gamma, e_1)$ in let $\tau = tc(\Gamma, e_2)$ in
if $\tau 1$ = unit then τ else error(...)

$tc(\Gamma$, if e_0 then e_1 else $e_2)$ =
let $\tau_i = tc(\Gamma, e_i)$ (for i=0,1,2) in
if τ_0 = int and $\tau_1 = \tau_2$ then τ_1 else error(...)

$tc(\Gamma, f(e_1,...,e_n))$ =
let $\tau_i = tc(\Gamma, e_i)$ (for i=1,...,n) in
if $(\tau_1, ..., \tau_n) \rightarrow \tau = \Gamma(f)$ for some τ
then τ else error(...)

$$\frac{\Gamma \vdash f: (\tau_1, ..., \tau_n)\rightarrow\tau \quad \Gamma \vdash e_i: \tau_i \quad (\text{for } i=1,...,n)}{\Gamma \vdash f(e_1,...,e_n): \tau}$$

型検査アルゴリズム

…

tc(Γ, let d in e) =

let Δ= tcdec(Γ, d) in let τ= tc((Γ,Δ), e) in τ

$$\frac{\Gamma \vdash d : \Delta \qquad \Gamma, \Delta \vdash e : \tau}{\Gamma \vdash \mathbf{let}\ d\ \mathbf{in}\ e : \tau}$$

型検査アルゴリズム

...

tc(Γ, let d in e) =

let Δ= tcdec(Γ, d) in let τ= tc((Γ,Δ), e) in τ

tcdec(Γ, var x:b := e) =

let τ= tc(Γ, e) in if τ=b then x:b else error(...)

$$\frac{\Gamma \vdash e : b}{\Gamma \vdash \mathrm{var}\ x{:}b := e \ : \ x{:}b}$$

型検査アルゴリズム

...

tc(Γ, let d in e) =

    let Δ= tcdec(Γ, d) in tc((Γ,Δ), e)

tcdec(Γ, var x:b := e) =

    let τ= tc(Γ, e) in if τ=b then x:b else error(...)

tcdec(Γ, fun f(x_1:b_1,...,x_n:b_n):b:=e) =

    let τ= tc((Γ, f:(b1,...,bn)→b, x_1:b_1,...,x_n:b_n), e) in

    if τ = b then f: (b_1,...,b_n)→b else error(...)

$$\frac{\Gamma,\ f:(b1,\dots,bn)\to b,\ x_1:b_1,\dots,x_n:b_n \vdash e: b}{\Gamma \vdash \mathrm{fun}\ f(x_1:b_1,\dots,x_n:b_n):b := e\ :\ f:(b_1,\dots,b_n)\to b}$$

アウトライン

- 型検査
  - 型システム
  - 型検査アルゴリズム
- 変数の参照関係等の明確化

型判断の拡張

$$\Gamma \vdash e: \tau \Rightarrow M$$

- Mは以下を除いてeと同じ
  - 異なる変数は別の変数名に

    e.g. var x:= 1; var x:= x+1; ...

    => var x':=1; var x:= x'+1; ...
  - 演算子のオーバーローディングを解消

    e.g. x+1 => x $+_{int}$ 1
- Γは、x_1: (y_1,τ_1), ..., x_n: (y_n,τ_n)の形に拡張
  (y_iは、名前をつけかえた後の変数名)

変換規則(変数、定数)

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

$$\frac{}{\Gamma \vdash n :\ int \Rightarrow n}$$

$$\frac{}{\Gamma \vdash s :\ string \Rightarrow s}$$

$$\frac{\Gamma \vdash e1 : b1 \Rightarrow M1 \quad \Gamma \vdash e2 : b2 \Rightarrow M2 \quad (b1, b2) \rightarrow b \in CType(op)}{\Gamma \vdash e1\ op\ e2 :\ b \Rightarrow M1\ op_{(b1,b2)\rightarrow b}\ M2}$$

変換規則(代入、制御)

$$\frac{\Gamma(x) = (y,\tau) \qquad \Gamma \vdash e : \tau \Rightarrow M}{\Gamma \vdash x := e : \mathrm{unit} \Rightarrow y := M}$$

$$\frac{\Gamma \vdash e_1 : \mathrm{unit} \Rightarrow M_1 \qquad \Gamma \vdash e_2 : \tau \Rightarrow M_2}{\Gamma \vdash e_1 \ ; \ e_2 : \tau \Rightarrow M_1 \ ; \ M_2}$$

$$\frac{\begin{array}{c}\Gamma \vdash e_0 : \mathrm{bool} \Rightarrow M_0 \\ \Gamma \vdash e_1 : \tau \Rightarrow M_1 \quad \Gamma \vdash e_2 : \tau \Rightarrow M_2\end{array}}{\begin{array}{c}\Gamma \vdash \mathrm{if}\ e_0\ \mathrm{then}\ e_1\ \mathrm{else}\ e_2 : \tau \\ \Rightarrow \mathrm{if}\ M_0\ \mathrm{then}\ M_1\ \mathrm{else}\ M_2\end{array}}$$

変換規則（関数呼び出し、let文）

$$\frac{\Gamma(f)=(g,\ (\tau_1,\ldots,\tau_n)\to\tau\ ) \qquad \Gamma \vdash e_i : \tau_i \Rightarrow M_i \quad (\text{for } i=1,\ldots,n)}{\Gamma \vdash f(e_1,\ldots,e_n) : \tau \Rightarrow g(M_1,\ldots,M_n) :}$$

$$\frac{\Gamma \vdash d : \Delta \Rightarrow D \qquad \Gamma, \Delta \vdash e : \tau \Rightarrow M}{\Gamma \vdash \text{let } d \text{ in } e : \tau \Rightarrow \text{let } D \text{ in } M}$$

変換規則(宣言)

$$\frac{\Gamma \vdash e: b \Rightarrow M \qquad y:\text{fresh}}{\Gamma \vdash \text{var } x{:}b := e : x{:}(y,b) \Rightarrow \text{var } y{:}b := M}$$

$$\frac{\Gamma,\ f{:}(g,(b_1,\ldots,b_n)\to b),\ x_1{:}(y_1,b_1),\ldots,x_n{:}(y_n,b_n) \vdash e: b \Rightarrow M \qquad g,\ y_1,\ldots,y_n :\text{fresh}}{\Gamma \vdash \text{fun } f(x_1{:}b_1,\ldots,x_n{:}b_n){:}b{:=}e : f{:}(g,(b_1,\ldots,b_n)\to b) \Rightarrow \text{fun } g(y_1{:}b_1,\ldots,y_n{:}b_n){:}b{:=}M}$$

型推論

- 入力：型宣言の入っていないプログラム
- 出力：型宣言つきのプログラム

- 推論方法
  - 各変数の型を表す（型）変数を用意
  - 型づけ規則に従って、型変数に関する制約を生成
  - 制約を解く（単一化）

型推論アルゴリズム

$\mathrm{tinf}(\Gamma, e) = (\tau\ ,\ C)$

C : $\Gamma \vdash e: \tau$ が成り立つための

Γ，τ中の型変数に関する（できれば必要）十分条件

$\mathrm{tinf}(\Gamma, x)$ = if $\Gamma(x)$ is defined then $(\Gamma(x), \emptyset)$

else error("x is undefined")

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

型推論アルゴリズム

tinf(Γ, x) = if Γ(x) is defined then (Γ(x), $\varnothing$)
else error("x is undefined")

tinf(Γ, n) = (int, $\varnothing$)

tinf(Γ, s) = (string, $\varnothing$)

$$\frac{}{\Gamma \vdash \mathbf{n}:\ \mathbf{int}}$$

$$\frac{}{\Gamma \vdash \mathbf{s}:\ \mathbf{string}}$$

型推論アルゴリズム

tinf(Γ, x) = if Γ(x) is defined then (Γ(x), $\emptyset$)
else error("x is undefined")

tinf(Γ, n) = (int, $\emptyset$)

tinf(Γ, s) = (string, $\emptyset$)

tinf(Γ, e1 op e2) =
let (τ1,C1) = tinf(Γ, e1) in let (τ2,C2) = tinf(Γ, e2) in
(α, C1$\cup$C2$\cup${CType(op)=(τ1,τ2)$\rightarrow\alpha$})

$$\frac{\Gamma \vdash e1{:}b1 \quad \Gamma \vdash e2{:}b2 \quad (b1,b2)\rightarrow b \in CType(op)}{\Gamma \vdash e1\ op\ e2{:}\ b}$$

型推論アルゴリズム

$\mathrm{tinf}(\Gamma, x)$ = if $\Gamma(x)$ is defined then $(\Gamma(x), \emptyset)$
else error("x is undefined")

$\mathrm{tinf}(\Gamma, n) = (\mathrm{int}, \emptyset)$

$\mathrm{tinf}(\Gamma, s) = (\mathrm{string}, \emptyset)$

$\mathrm{tinf}(\Gamma, e1\ \mathrm{op}\ e2) =$

let $(\tau 1, C1) = \mathrm{tinf}(\Gamma, e1)$ in let $(\tau 2, C2) = \mathrm{tinf}(\Gamma, e2)$ in

$(\alpha, C1 \cup C2 \cup \{CType(op)=(\tau 1, \tau 2) \rightarrow \alpha\})$

$\mathrm{tinf}(\Gamma, x := e)$ = let $(\tau, C) = \mathrm{tinf}(\Gamma, e)$ in

$(\mathrm{unit}, C \cup \{\tau = \Gamma(x)\})$

$$\frac{\Gamma(x) = \tau \qquad \Gamma \vdash e : \tau}{\Gamma \vdash x := e : \mathrm{unit}}$$

型推論アルゴリズム

…

$\mathrm{tinf}(\Gamma, e_1\ ;\ e_2) = \mathrm{let}\ (\tau1, C1) = \mathrm{tinf}(\Gamma, e_1)\ \mathrm{in}$

$\mathrm{let}\ (\tau2, C2) = \mathrm{tinf}(\Gamma, e_2)\ \mathrm{in}$

$(\tau2, C1 \cup C2 \cup \{\tau1 = \mathrm{unit}\})$

$$\frac{\Gamma \vdash e_1 : \mathrm{unit} \qquad \Gamma \vdash e_2 : \tau}{\Gamma \vdash e_1\ ;\ e_2 : \tau}$$

型推論アルゴリズム

...

$\mathrm{tinf}(\Gamma, e_1\ ;\ e_2)$ = let $(\tau 1, C1)$= $\mathrm{tinf}(\Gamma, e_1)$ in
let $(\tau 2, C2)$= $\mathrm{tinf}(\Gamma, e_2)$ in
$(\tau 2, C1 \cup C2 \cup \{\tau 1 = \mathrm{unit}\})$

$\mathrm{tinf}(\Gamma$, if e_0 then e_1 else $e_2)$ =
let $(\tau i, Ci) = \mathrm{tinf}(\Gamma, e_i)$ (for i=0,1,2) in
$(\tau 1,\ C0 \cup C1 \cup C2 \cup \{\tau 0 = \mathrm{int},\ \tau 1 = \tau 2\})$

$$\frac{\Gamma \vdash e_0 : \mathrm{int} \quad \Gamma \vdash e_1 : \tau \quad \Gamma \vdash e_2 : \tau}{\Gamma \vdash \text{if } e_0 \text{ then } e_1 \text{ else } e_2 : \tau}$$

型推論アルゴリズム

...

$\mathrm{tinf}(\Gamma, e_1\ ;\ e_2) = \mathrm{let}\ (\tau 1, C1) = \mathrm{tinf}(\Gamma, e_1)\ \mathrm{in}$

$\qquad \mathrm{let}\ (\tau 2, C2) = \mathrm{tinf}(\Gamma, e_2)\ \mathrm{in}$

$\qquad (\tau 2, C1 \cup C2 \cup \{\tau 1 = \mathrm{unit}\})$

$\mathrm{tinf}(\Gamma, \mathrm{if}\ e_0\ \mathrm{then}\ e_1\ \mathrm{else}\ e_2) =$

$\qquad \mathrm{let}\ (\tau i, Ci) = \mathrm{tinf}(\Gamma, e_i)\ (\mathrm{for}\ i=0,1,2)\ \mathrm{in}$

$\qquad (\tau 1,\ C0 \cup C1 \cup C2 \cup \{\tau 0 = \mathrm{int},\ \tau 1 = \tau 2\})$

$\mathrm{tinf}(\Gamma, f(e_1, \ldots, e_n)) =$

$\qquad \mathrm{let}\ (\tau i, Ci) = \mathrm{tinf}(\Gamma, e_i)\ (\mathrm{for}\ i=1,\ldots,n)\ \mathrm{in}$

$\qquad (\alpha,\ C1 \cup \ldots \cup Cn \cup \{(\tau_1, \ldots, \tau_n) \rightarrow \alpha = \Gamma(f)\})$

$$\frac{\Gamma \vdash f: (\tau_1, \ldots, \tau_n) \rightarrow \tau \quad \Gamma \vdash e_i : \tau_i \quad (\mathrm{for}\ i=1,\ldots,n)}{\Gamma \vdash f(e_1, \ldots, e_n): \tau}$$

型推論アルゴリズム

...

```
tinf(Γ, let d in e) =
    let (Δ,C1)= tinfdec(Γ, d) in
    let (τ,C2)=tinf((Γ,Δ), e) in (τ, C1∪C2)
```

$$\frac{\Gamma \vdash d : \Delta \qquad \Gamma, \Delta \vdash e : \tau}{\Gamma \vdash \mathbf{let}\ d\ \mathbf{in}\ e : \tau}$$

型推論アルゴリズム

...

tinf(Γ, let d in e) =

  let (Δ,C1)= tinfdec(Γ, d) in

  let (τ,C2)=tinf((Γ,Δ), e) in (τ, C1∪C2)

tinfdec(Γ, var x := e) =

  let (τ,C) = tinf(Γ, e) in (x:τ,C)

$$\frac{\Gamma \vdash e : b}{\Gamma \vdash \mathtt{var}\ x{:}b := e \ : \ x{:}b}$$

型推論アルゴリズム

...

tinf(Γ, let d in e) =

  let (Δ,C1)= tinfdec(Γ, d) in

  let (τ,C2)=tinf((Γ,Δ), e) in (τ, C1$\cup$C2)

tinfdec(Γ, var x := e) =

  let (τ,C) = tinf(Γ, e) in (x:τ,C)

tinfdec(Γ, fun f(x_1,...,x_n):=e) =

 let (τ,C)=tinf((Γ,f:(α1,...,αn)$\rightarrow\alpha$, x_1:α_1,...,x_n:α_n), e) in

  (f:(α1,...,αn)$\rightarrow\alpha$, C $\cup$ {τ = α})

$$\frac{\Gamma,\ f:(b1,\ldots,bn)\rightarrow b,\ x_1:b_1,\ldots,x_n:b_n \vdash e:\ b}{\Gamma \vdash \mathrm{fun}\ f(x_1:b_1,\ldots,x_n:b_n):b := e\ :\ f:(b_1,\ldots,b_n)\rightarrow b}$$

型推論

- 入力：型宣言の入っていないプログラム
- 出力：型宣言つきのプログラム

- 推論方法
  - 各変数の型を表す（型）変数を用意
  - 型づけ規則に従って、型変数に関する制約を生成
  - **制約を解く（単一化）**

時間があれば黒板で

型推論の例

<黒板で>

```
let var x := 1 in
let fun f(y) := y in
  f (x)
```