

Programming Languages and Compilers (CS 421)

Elsa L Gunter
2112 SC, UIUC

<http://courses.engr.illinois.edu/cs421>

Based in part on slides by Mattox Beckman, as updated by Vikram Adve and Gul Agha

11/2/19

1

Support for Polymorphic Types

- Monomorphic Types (τ):
 - Basic Types: `int`, `bool`, `float`, `string`, `unit`, ...
 - Type Variables: $\alpha, \beta, \gamma, \delta, \epsilon$
 - Compound Types: $\alpha \rightarrow \beta$, `int * string`, `bool list`, ...
- Polymorphic Types:
 - Monomorphic types τ
 - Universally quantified monomorphic types
 - $\forall \alpha_1, \dots, \alpha_n. \tau$
 - Can think of τ as same as $\forall. \tau$

11/2/19

2

Example: $\{\} \vdash 2 = 3 : \text{bool}$

$(=) : \text{All } 'a. 'a \rightarrow 'a \rightarrow \text{bool}$

Instance: $'a \rightarrow \text{int}$

```

-----Const
{} |- (=) : -----Const
int -> int -> bool   {} |- 2:int
-----App           -----Const
{} |- (=) 2 : int -> bool   {} |- 3 : int
-----App
{} |- 2 = 3 : bool
    
```

11/2/19

3

Example FreeVars Calculations

- $\text{Vars}('a \rightarrow (\text{int} \rightarrow 'b) \rightarrow 'a) = \{ 'a, 'b \}$
- $\text{FreeVars} (\text{All } 'b. 'a \rightarrow (\text{int} \rightarrow 'b) \rightarrow 'a) = \{ 'a, 'b \} - \{ 'b \} = \{ 'a \}$
- $\text{FreeVars } \{x : \text{All } 'b. 'a \rightarrow (\text{int} \rightarrow 'b) \rightarrow 'a, \text{id} : \text{All } 'c. 'c \rightarrow 'c, y : \text{All } 'c. 'a \rightarrow 'b \rightarrow 'c\} = \{ 'a \} \cup \{ \} \cup \{ 'a, 'b \} = \{ 'a, 'b \}$

11/2/19

4

Support for Polymorphic Types

- Typing Environment Γ supplies polymorphic types (which will often just be monomorphic) for variables
- Free variables of monomorphic type just type variables that occur in it
 - Write $\text{FreeVars}(\tau)$
- Free variables of polymorphic type removes variables that are universally quantified
 - $\text{FreeVars}(\forall \alpha_1, \dots, \alpha_n. \tau) = \text{FreeVars}(\tau) - \{ \alpha_1, \dots, \alpha_n \}$
- $\text{FreeVars}(\Gamma) =$ all FreeVars of types in range of Γ

11/2/19

5

Monomorphic to Polymorphic

- Given:
 - type environment Γ
 - monomorphic type τ
 - τ shares type variables with Γ
- Want most polymorphic type for τ that doesn't break sharing type variables with Γ
- $\text{Gen}(\tau, \Gamma) = \forall \alpha_1, \dots, \alpha_n. \tau$ where $\{ \alpha_1, \dots, \alpha_n \} = \text{freeVars}(\tau) - \text{freeVars}(\Gamma)$

11/2/19

6

Polymorphic Typing Rules

- A *type judgement* has the form $\Gamma \vdash \text{exp} : \tau$
 - Γ uses **polymorphic** types
 - τ still monomorphic
- Most rules stay same (except use more general typing environments)
- Rules that change:
 - Variables
 - Let and Let Rec
 - Allow polymorphic constants
- Worth noting functions again

11/2/19

7

Two Problems

- Type checking
 - Question: Does exp. e have type τ in env Γ ?
 - Answer: Yes / No
 - Method: Type **derivation**
- Typability
 - Question Does exp. e have **some type** in env. Γ ?
If so, what is it?
 - Answer: Type τ / error
 - Method: Type **inference**

11/2/19

8

Type Inference - Outline

- Begin by assigning a type variable as the type of the whole expression
- Decompose the expression into component expressions
- Use typing rules to generate constraints on components and whole
- Recursively find substitution that solves typing judgment of first subcomponent
- Apply substitution to next subcomponent and find substitution solving it; compose with first, etc.
- Apply comp of all substitution to orig. type var. to get answer

11/2/19

9

Type Inference - Example

- What type can we give to $(\text{fun } x \rightarrow \text{fun } f \rightarrow f (f x))$
- Start with a type variable and then look at the way the term is constructed

11/2/19

10

Type Inference - Example

- First approximate:

$$\{ \} \vdash (\text{fun } x \rightarrow \text{fun } f \rightarrow f (f x)) : \alpha$$
- Second approximate: use fun rule

$$\frac{\{x : \beta\} \vdash (\text{fun } f \rightarrow f (f x)) : \gamma}{\{ \} \vdash (\text{fun } x \rightarrow \text{fun } f \rightarrow f (f x)) : \alpha}$$
- Remember constraint $\alpha \equiv (\beta \rightarrow \gamma)$

11/2/19

11

Type Inference - Example

- Third approximate: use fun rule

$$\frac{\frac{\{f : \delta ; x : \beta\} \vdash f (f x) : \epsilon}{\{x : \beta\} \vdash (\text{fun } f \rightarrow f (f x)) : \gamma}}{\{ \} \vdash (\text{fun } x \rightarrow \text{fun } f \rightarrow f (f x)) : \alpha}$$
- $\alpha \equiv (\beta \rightarrow \gamma); \gamma \equiv (\delta \rightarrow \epsilon)$

11/2/19

12

Type Inference - Example

- Fourth approximate: use app rule

$$\frac{\frac{\frac{\{f:\delta; x:\beta\}|- f : \varphi \rightarrow \varepsilon \quad \{f:\delta; x:\beta\}|- f x : \varphi}{\{f : \delta ; x : \beta \} |- (f (f x)) : \varepsilon}}{\{x : \beta \} |- (fun f \rightarrow f (f x)) : \gamma}}{\{ \} |- (fun x \rightarrow fun f \rightarrow f (f x)) : \alpha}$$

- $\alpha \equiv (\beta \rightarrow \gamma); \gamma \equiv (\delta \rightarrow \varepsilon)$

11/2/19

13

Type Inference - Example

- Fifth approximate: use var rule, get constraint $\delta \equiv \varphi \rightarrow \varepsilon$, Solve with same
- Apply to next sub-proof

$$\frac{\frac{\frac{\frac{\{f:\delta; x:\beta\}|- f : \varphi \rightarrow \varepsilon \quad \{f:\delta; x:\beta\}|- f x : \varphi}{\{f : \delta ; x : \beta \} |- (f (f x)) : \varepsilon}}{\{x : \beta \} |- (fun f \rightarrow f (f x)) : \gamma}}{\{ \} |- (fun x \rightarrow fun f \rightarrow f (f x)) : \alpha}}$$

- $\alpha \equiv (\beta \rightarrow \gamma); \gamma \equiv (\delta \rightarrow \varepsilon)$

11/2/19

14

Type Inference - Example

- Current subst: $\{\delta \equiv \varphi \rightarrow \varepsilon\}$

$$\frac{\frac{\frac{\frac{\dots \quad \{f:\varphi \rightarrow \varepsilon; x:\beta\}|- f x : \varphi}{\{f : \delta ; x : \beta \} |- (f (f x)) : \varepsilon}}{\{x : \beta \} |- (fun f \rightarrow f (f x)) : \gamma}}{\{ \} |- (fun x \rightarrow fun f \rightarrow f (f x)) : \alpha}}$$

- $\alpha \equiv (\beta \rightarrow \gamma); \gamma \equiv (\delta \rightarrow \varepsilon)$

11/2/19

15

Type Inference - Example

- Current subst: $\{\delta \equiv \varphi \rightarrow \varepsilon\}$ Use App Rule

$$\frac{\frac{\frac{\frac{\frac{\{f:\varphi \rightarrow \varepsilon; x:\beta\}|- f:\zeta \rightarrow \varphi \quad \{f:\varphi \rightarrow \varepsilon; x:\beta\}|- x:\zeta}{\dots \quad \{f:\varphi \rightarrow \varepsilon; x:\beta\}|- f x : \varphi}}{\{f : \delta ; x : \beta \} |- (f (f x)) : \varepsilon}}{\{x : \beta \} |- (fun f \rightarrow f (f x)) : \gamma}}{\{ \} |- (fun x \rightarrow fun f \rightarrow f (f x)) : \alpha}}$$

- $\alpha \equiv (\beta \rightarrow \gamma); \gamma \equiv (\delta \rightarrow \varepsilon)$

11/2/19

16

Type Inference - Example

- Current subst: $\{\delta \equiv \varphi \rightarrow \varepsilon\}$
- Var rule: Solve $\zeta \rightarrow \varphi \equiv \varphi \rightarrow \varepsilon$ **Unification**

$$\frac{\frac{\frac{\frac{\frac{\{f:\varphi \rightarrow \varepsilon; x:\beta\}|- f:\zeta \rightarrow \varphi \quad \{f:\varphi \rightarrow \varepsilon; x:\beta\}|- x:\zeta}{\dots \quad \{f:\varphi \rightarrow \varepsilon; x:\beta\}|- f x : \varphi}}{\{f : \delta ; x : \beta \} |- (f (f x)) : \varepsilon}}{\{x : \beta \} |- (fun f \rightarrow f (f x)) : \gamma}}{\{ \} |- (fun x \rightarrow fun f \rightarrow f (f x)) : \alpha}}$$

- $\alpha \equiv (\beta \rightarrow \gamma); \gamma \equiv (\delta \rightarrow \varepsilon)$

11/2/19

17

Type Inference - Example

- Current subst: $\{\zeta \equiv \varepsilon, \varphi \equiv \varepsilon\} \circ \{\delta \equiv \varphi \rightarrow \varepsilon\}$
- Var rule: Solve $\zeta \rightarrow \varphi \equiv \varphi \rightarrow \varepsilon$ **Unification**

$$\frac{\frac{\frac{\frac{\frac{\{f:\varphi \rightarrow \varepsilon; x:\beta\}|- f:\zeta \rightarrow \varphi \quad \{f:\varphi \rightarrow \varepsilon; x:\beta\}|- x:\zeta}{\dots \quad \{f:\varphi \rightarrow \varepsilon; x:\beta\}|- f x : \varphi}}{\{f : \delta ; x : \beta \} |- (f (f x)) : \varepsilon}}{\{x : \beta \} |- (fun f \rightarrow f (f x)) : \gamma}}{\{ \} |- (fun x \rightarrow fun f \rightarrow f (f x)) : \alpha}}$$

- $\alpha \equiv (\beta \rightarrow \gamma); \gamma \equiv (\delta \rightarrow \varepsilon)$

11/2/19

18

Type Inference - Example

- Current subst: $\{\zeta \equiv \varepsilon, \varphi \equiv \varepsilon, \delta \equiv \varepsilon \rightarrow \varepsilon\}$
- Apply to next sub-proof

$$\frac{\dots \quad \frac{\{f:\varepsilon \rightarrow \varepsilon; x:\beta\} \vdash x:\varepsilon}{\dots \quad \{f:\varphi \rightarrow \varepsilon; x:\beta\} \vdash f x:\varphi}}{\{f:\delta; x:\beta\} \vdash (f(f x)):\varepsilon}}{\{x:\beta\} \vdash (\text{fun } f \rightarrow f(f x)):\gamma}}{\{\} \vdash (\text{fun } x \rightarrow \text{fun } f \rightarrow f(f x)):\alpha}$$

- $\alpha \equiv (\beta \rightarrow \gamma); \gamma \equiv (\delta \rightarrow \varepsilon)$

11/2/19

19

Type Inference - Example

- Current subst: $\{\zeta \equiv \varepsilon, \varphi \equiv \varepsilon, \delta \equiv \varepsilon \rightarrow \varepsilon\}$
- Var rule: $\varepsilon \equiv \beta$

$$\frac{\dots \quad \frac{\{f:\varepsilon \rightarrow \varepsilon; x:\beta\} \vdash x:\varepsilon}{\dots \quad \{f:\varphi \rightarrow \varepsilon; x:\beta\} \vdash f x:\varphi}}{\{f:\delta; x:\beta\} \vdash (f(f x)):\varepsilon}}{\{x:\beta\} \vdash (\text{fun } f \rightarrow f(f x)):\gamma}}{\{\} \vdash (\text{fun } x \rightarrow \text{fun } f \rightarrow f(f x)):\alpha}$$

- $\alpha \equiv (\beta \rightarrow \gamma); \gamma \equiv (\delta \rightarrow \varepsilon)$

11/2/19

20

Type Inference - Example

- Current subst: $\{\varepsilon \equiv \beta\} \circ \{\zeta \equiv \varepsilon, \varphi \equiv \varepsilon, \delta \equiv \varepsilon \rightarrow \varepsilon\}$
- Solves subproof; return one layer

$$\frac{\dots \quad \frac{\{f:\varepsilon \rightarrow \varepsilon; x:\beta\} \vdash x:\varepsilon}{\dots \quad \{f:\varphi \rightarrow \varepsilon; x:\beta\} \vdash f x:\varphi}}{\{f:\delta; x:\beta\} \vdash (f(f x)):\varepsilon}}{\{x:\beta\} \vdash (\text{fun } f \rightarrow f(f x)):\gamma}}{\{\} \vdash (\text{fun } x \rightarrow \text{fun } f \rightarrow f(f x)):\alpha}$$

- $\alpha \equiv (\beta \rightarrow \gamma); \gamma \equiv (\delta \rightarrow \varepsilon)$

11/2/19

21

Type Inference - Example

- Current subst: $\{\varepsilon \equiv \beta, \zeta \equiv \beta, \varphi \equiv \beta, \delta \equiv \beta \rightarrow \beta\}$
- Solves this subproof; return one layer

$$\frac{\dots \quad \frac{\{f:\varphi \rightarrow \varepsilon; x:\beta\} \vdash f x:\varphi}{\{f:\delta; x:\beta\} \vdash (f(f x)):\varepsilon}}{\{x:\beta\} \vdash (\text{fun } f \rightarrow f(f x)):\gamma}}{\{\} \vdash (\text{fun } x \rightarrow \text{fun } f \rightarrow f(f x)):\alpha}$$

- $\alpha \equiv (\beta \rightarrow \gamma); \gamma \equiv (\delta \rightarrow \varepsilon)$

11/2/19

22

Type Inference - Example

- Current subst: $\{\varepsilon \equiv \beta, \zeta \equiv \beta, \varphi \equiv \beta, \delta \equiv \beta \rightarrow \beta\}$
- Need to satisfy constraint $\gamma \equiv (\delta \rightarrow \varepsilon)$, given subst, becomes: $\gamma \equiv ((\beta \rightarrow \beta) \rightarrow \beta)$

$$\frac{\dots \quad \frac{\{f:\delta; x:\beta\} \vdash (f(f x)):\varepsilon}{\{x:\beta\} \vdash (\text{fun } f \rightarrow f(f x)):\gamma}}{\{\} \vdash (\text{fun } x \rightarrow \text{fun } f \rightarrow f(f x)):\alpha}$$

- $\alpha \equiv (\beta \rightarrow \gamma); \gamma \equiv (\delta \rightarrow \varepsilon)$

11/2/19

23

Type Inference - Example

- Current subst: $\{\gamma \equiv ((\beta \rightarrow \beta) \rightarrow \beta), \varepsilon \equiv \beta, \zeta \equiv \beta, \varphi \equiv \beta, \delta \equiv \beta \rightarrow \beta\}$
- Solves subproof; return one layer

$$\frac{\dots \quad \frac{\{f:\delta; x:\beta\} \vdash (f(f x)):\varepsilon}{\{x:\beta\} \vdash (\text{fun } f \rightarrow f(f x)):\gamma}}{\{\} \vdash (\text{fun } x \rightarrow \text{fun } f \rightarrow f(f x)):\alpha}$$

- $\alpha \equiv (\beta \rightarrow \gamma); \gamma \equiv (\delta \rightarrow \varepsilon)$

11/2/19

24

Type Inference - Example

- Current subst:

$\{\gamma \equiv ((\beta \rightarrow \beta) \rightarrow \beta), \varepsilon \equiv \beta, \zeta \equiv \beta, \varphi \equiv \beta, \delta \equiv \beta \rightarrow \beta\}$

- Need to satisfy constraint $\alpha \equiv (\beta \rightarrow \gamma)$
given subst: $\alpha \equiv (\beta \rightarrow ((\beta \rightarrow \beta) \rightarrow \beta))$

$$\frac{\dots}{\frac{\{x : \beta\} \vdash (\text{fun } f \rightarrow f (f x)) : \gamma}{\{ \} \vdash (\text{fun } x \rightarrow \text{fun } f \rightarrow f (f x)) : \alpha}}$$

- $\alpha \equiv (\beta \rightarrow \gamma)$;

11/2/19

25

Type Inference - Example

- Current subst:

$\{\alpha \equiv (\beta \rightarrow ((\beta \rightarrow \beta) \rightarrow \beta)),$

$\gamma \equiv ((\beta \rightarrow \beta) \rightarrow \beta), \varepsilon \equiv \beta, \zeta \equiv \beta, \varphi \equiv \beta, \delta \equiv \beta \rightarrow \beta\}$

- Solves subproof; return on layer

$$\frac{\dots}{\frac{\{x : \beta\} \vdash (\text{fun } f \rightarrow f (f x)) : \gamma}{\{ \} \vdash (\text{fun } x \rightarrow \text{fun } f \rightarrow f (f x)) : \alpha}}$$

11/2/19

26

Type Inference - Example

- Current subst:

$\{\alpha \equiv (\beta \rightarrow ((\beta \rightarrow \beta) \rightarrow \beta)),$

$\gamma \equiv ((\beta \rightarrow \beta) \rightarrow \beta), \varepsilon \equiv \beta, \zeta \equiv \beta, \varphi \equiv \beta, \delta \equiv \beta \rightarrow \beta\}$

- Done: $\alpha \equiv (\beta \rightarrow ((\beta \rightarrow \beta) \rightarrow \beta))$

$$\{ \} \vdash (\text{fun } x \rightarrow \text{fun } f \rightarrow f (f x)) : \alpha$$

11/2/19

27

Type Inference Algorithm

Let $\text{infer}(\Gamma, e, \tau) = \sigma$

- Γ is a typing environment (giving polymorphic types to expression variables)
- e is an expression
- τ is a type (with type variables),
- σ is a substitution of types for type variables
- Idea: σ is the constraints on type variables necessary for $\Gamma \vdash e : \tau$
- Should have $\sigma(\Gamma) \vdash e : \sigma(\tau)$ valid

11/2/19

28

Type Inference Algorithm

$\text{infer}(\Gamma, \text{exp}, \tau) =$

- Case exp of
 - Var $v \rightarrow$ return $\text{Unify}\{\tau \equiv \text{freshInstance}(\Gamma(v))\}$
 - Replace all quantified type vars by fresh ones
 - Const $c \rightarrow$ return $\text{Unify}\{\tau \equiv \text{freshInstance } \varphi\}$
where $\Gamma \vdash c : \varphi$ by the constant rules
 - fun $x \rightarrow e \rightarrow$
 - Let α, β be fresh variables
 - Let $\sigma = \text{infer}(\{x: \alpha\} + \Gamma, e, \beta)$
 - Return $\text{Unify}\{\{\sigma(\tau) \equiv \sigma(\alpha \rightarrow \beta)\}\} \circ \sigma$

11/2/19

29

Type Inference Algorithm (cont)

- Case exp of

- App ($e_1 e_2$) \rightarrow

- Let α be a fresh variable
- Let $\sigma_1 = \text{infer}(\Gamma, e_1, \alpha \rightarrow \tau)$
- Let $\sigma_2 = \text{infer}(\sigma_1(\Gamma), e_2, \sigma_1(\alpha))$
- Return $\sigma_2 \circ \sigma_1$

11/2/19

30

Type Inference Algorithm (cont)

- Case *exp* of
 - If e_1 then e_2 else e_3 -->
 - Let $\sigma_1 = \text{infer}(\Gamma, e_1, \text{bool})$
 - Let $\sigma_2 = \text{infer}(\sigma_1\Gamma, e_2, \sigma_1(\tau))$
 - Let $\sigma_3 = \text{infer}(\sigma_2 \circ \sigma_1(\Gamma), e_3, \sigma_2 \circ \sigma_1(\tau))$
 - Return $\sigma_3 \circ \sigma_2 \circ \sigma_1$

11/2/19

31

Type Inference Algorithm (cont)

- Case *exp* of
 - let $x = e_1$ in e_2 -->
 - Let α be a fresh variable
 - Let $\sigma_1 = \text{infer}(\Gamma, e_1, \alpha)$
 - Let $\sigma_2 =$
 $\text{infer}(\{x:\text{GEN}(\sigma_1(\Gamma), \sigma_1(\alpha))\} + \sigma_1(\Gamma),$
 $e_2, \sigma_1(\tau))$
 - Return $\sigma_2 \circ \sigma_1$

11/2/19

32

Type Inference Algorithm (cont)

- Case *exp* of
 - let rec $x = e_1$ in e_2 -->
 - Let α be a fresh variable
 - Let $\sigma_1 = \text{infer}(\{x:\alpha\} + \Gamma, e_1, \alpha)$
 - Let $\sigma_2 = \text{infer}(\{x:\text{GEN}(\sigma_1(\Gamma), \sigma_1(\alpha))\}$
 $+ \sigma_1(\Gamma)\}, e_2, \sigma_1(\tau))$
 - Return $\sigma_2 \circ \sigma_1$

11/2/19

33

Type Inference Algorithm (cont)

- To infer a type, introduce *type_of*
- Let α be a fresh variable
- *type_of* $(\Gamma, e) =$
 - Let $\sigma = \text{infer}(\Gamma, e, \alpha)$
 - Return $\sigma(\alpha)$
- Need an algorithm for Unif

11/2/19

34

Background for Unification

- Terms made from **constructors** and **variables** (for the simple first order case)
- Constructors may be **applied** to arguments (other terms) to make new terms
- Variables and constructors with no arguments are base cases
- Constructors applied to different number of arguments (**arity**) considered different
- **Substitution** of terms for variables

11/2/19

35

Simple Implementation Background

```
type term = Variable of string
          | Const of (string * term list)
let x = Variable "a";; let tm = Const ("2",[]);;

let rec subst var_name residue term =
  match term with Variable name ->
    if var_name = name then residue else term
  | Const (c, tys) ->
    Const (c, List.map (subst var_name residue)
                    tys);;
```

11/2/19

36

Unification Problem

Given a set of pairs of terms (“equations”)

$$\{(s_1, t_1), (s_2, t_2), \dots, (s_n, t_n)\}$$

(the *unification problem*) does there exist a substitution σ (the *unification solution*) of terms for variables such that

$$\sigma(s_i) = \sigma(t_i),$$

for all $i = 1, \dots, n$?

11/2/19

37

Uses for Unification

- Type Inference and type checking
- Pattern matching as in OCaml
 - Can use a simplified version of algorithm
- Logic Programming - Prolog
- Simple parsing

11/2/19

38

Unification Algorithm

- Let $S = \{(s_1 = t_1), (s_2 = t_2), \dots, (s_n = t_n)\}$ be a unification problem.
- Case $S = \{\}$: $\text{Unif}(S) = \text{Identity function}$ (i.e., no substitution)
- Case $S = \{(s, t)\} \cup S'$: Four main steps

11/2/19

39

Unification Algorithm

- **Delete:** if $s = t$ (they are the same term) then $\text{Unif}(S) = \text{Unif}(S')$
- **Decompose:** if $s = f(q_1, \dots, q_m)$ and $t = f(r_1, \dots, r_m)$ (same f , same m !), then $\text{Unif}(S) = \text{Unif}(\{(q_1, r_1), \dots, (q_m, r_m)\} \cup S')$
- **Orient:** if $t = x$ is a variable, and s is not a variable, $\text{Unif}(S) = \text{Unif}(\{(x = s)\} \cup S')$

11/2/19

40

Unification Algorithm

- **Eliminate:** if $s = x$ is a variable, and x does not occur in t (the occurs check), then
 - Let $\varphi = \{x \rightarrow t\}$
 - $\text{Unif}(S) = \text{Unif}(\varphi(S')) \circ \{x \rightarrow t\}$
 - Let $\psi = \text{Unif}(\varphi(S'))$
 - $\text{Unif}(S) = \{x \rightarrow \psi(t)\} \circ \psi$
 - Note: $\{x \rightarrow a\} \circ \{y \rightarrow b\} = \{y \rightarrow (\{x \rightarrow a\}(b))\} \circ \{x \rightarrow a\}$ if y not in a

11/2/19

41

Tricks for Efficient Unification

- Don't return substitution, rather do it incrementally
- Make substitution be constant time
 - Requires implementation of terms to use mutable structures (or possibly lazy structures)
 - We won't discuss these

11/2/19

42

Example

- x, y, z variables, f, g constructors
- Unify $\{(f(x) = f(g(f(z), y))), (g(y, y) = x)\} = ?$

11/2/19

43

Example

- x, y, z variables, f, g constructors
- $S = \{(f(x) = f(g(f(z), y))), (g(y, y) = x)\}$ is nonempty
- Unify $\{(f(x) = f(g(f(z), y))), (g(y, y) = x)\} = ?$

11/2/19

44

Example

- x, y, z variables, f, g constructors
- Pick a pair: $(g(y, y) = x)$
- Unify $\{(f(x) = f(g(f(z), y))), (g(y, y) = x)\} = ?$

11/2/19

45

Example

- x, y, z variables, f, g constructors
- Pick a pair: $(g(y, y) = x)$
- Orient: $(x = g(y, y))$
- Unify $\{(f(x) = f(g(f(z), y))), (g(y, y) = x)\} =$
Unify $\{(f(x) = f(g(f(z), y))), (x = g(y, y))\}$
by Orient

11/2/19

46

Example

- x, y, z variables, f, g constructors
- Unify $\{(f(x) = f(g(f(z), y))), (x = g(y, y))\} = ?$

11/2/19

47

Example

- x, y, z variables, f, g constructors
- $\{(f(x) = f(g(f(z), y))), (x = g(y, y))\}$ is nonempty
- Unify $\{(f(x) = f(g(f(z), y))), (x = g(y, y))\} = ?$

11/2/19

48

Example

- x, y, z variables, f, g constructors
- Pick a pair: $(x = g(y, y))$

- Unify $\{(f(x) = f(g(f(z), y))), (x = g(y, y))\} = ?$

11/2/19

49

Example

- x, y, z variables, f, g constructors
- Pick a pair: $(x = g(y, y))$
- Eliminate x with substitution $\{x \rightarrow g(y, y)\}$
 - Check: x not in $g(y, y)$
- Unify $\{(f(x) = f(g(f(z), y))), (x = g(y, y))\} = ?$

11/2/19

50

Example

- x, y, z variables, f, g constructors
- Pick a pair: $(x = g(y, y))$
- Eliminate x with substitution $\{x \rightarrow g(y, y)\}$

- Unify $\{(f(x) = f(g(f(z), y))), (x = g(y, y))\} =$
Unify $\{(f(g(y, y)) = f(g(f(z), y)))\}$
 - $\{x \rightarrow g(y, y)\}$

11/2/19

51

Example

- x, y, z variables, f, g constructors

- Unify $\{(f(g(y, y)) = f(g(f(z), y)))\}$
 - $\{x \rightarrow g(y, y)\} = ?$

11/2/19

52

Example

- x, y, z variables, f, g constructors
- $\{(f(g(y, y)) = f(g(f(z), y)))\}$ is non-empty

- Unify $\{(f(g(y, y)) = f(g(f(z), y)))\}$
 - $\{x \rightarrow g(y, y)\} = ?$

11/2/19

53

Example

- x, y, z variables, f, g constructors
- Pick a pair: $(f(g(y, y)) = f(g(f(z), y)))$

- Unify $\{(f(g(y, y)) = f(g(f(z), y)))\}$
 - $\{x \rightarrow g(y, y)\} = ?$

11/2/19

54

Example

- x, y, z variables, f, g constructors
- Pick a pair: $(f(g(y, y)) = f(g(f(z), y)))$
- Decompose: $(f(g(y, y)) = f(g(f(z), y)))$ becomes $\{(g(y, y) = g(f(z), y))\}$
- Unify $\{(f(g(y, y)) = f(g(f(z), y)))\}$
 - $\{x \rightarrow g(y, y)\} =$
- Unify $\{(g(y, y) = g(f(z), y))\} \circ \{x \rightarrow g(y, y)\}$

11/2/19

55

Example

- x, y, z variables, f, g constructors
- $\{(g(y, y) = g(f(z), y))\}$ is non-empty
- Unify $\{(g(y, y) = g(f(z), y))\}$
 - $\{x \rightarrow g(y, y)\} = ?$

11/2/19

56

Example

- x, y, z variables, f, g constructors
- Pick a pair: $(g(y, y) = g(f(z), y))$
- Unify $\{(g(y, y) = g(f(z), y))\}$
 - $\{x \rightarrow g(y, y)\} = ?$

11/2/19

57

Example

- x, y, z variables, f, g constructors
- Pick a pair: $(f(g(y, y)) = f(g(f(z), y)))$
- Decompose: $(g(y, y) = g(f(z), y))$ becomes $\{(y = f(z)); (y = y)\}$
- Unify $\{(g(y, y) = g(f(z), y))\} \circ \{x \rightarrow g(y, y)\} =$
Unify $\{(y = f(z)); (y = y)\} \circ \{x \rightarrow g(y, y)\}$

11/2/19

58

Example

- x, y, z variables, f, g constructors
- Unify $\{(y = f(z)); (y = y)\} \circ \{x \rightarrow g(y, y)\} = ?$

11/2/19

59

Example

- x, y, z variables, f, g constructors
- $\{(y = f(z)); (y = y)\} \circ \{x \rightarrow g(y, y)\}$ is non-empty
- Unify $\{(y = f(z)); (y = y)\} \circ \{x \rightarrow g(y, y)\} = ?$

11/2/19

60

Example

- x, y, z variables, f, g constructors
- Pick a pair: $(y = f(z))$

- Unify $\{(y = f(z)); (y = y)\} \circ \{x \rightarrow g(y, y)\} = ?$

11/2/19

61

Example

- x, y, z variables, f, g constructors
- Pick a pair: $(y = f(z))$
- Eliminate y with $\{y \rightarrow f(z)\}$
- Unify $\{(y = f(z)); (y = y)\} \circ \{x \rightarrow g(y, y)\} =$
Unify $\{(f(z) = f(z))\}$
 - $\{y \rightarrow f(z)\} \circ \{x \rightarrow g(y, y)\} =$Unify $\{(f(z) = f(z))\}$
 - $\{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\}$

11/2/19

62

Example

- x, y, z variables, f, g constructors

- Unify $\{(f(z) = f(z))\}$
 - $\{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\} = ?$

11/2/19

63

Example

- x, y, z variables, f, g constructors
- $\{(f(z) = f(z))\}$ is non-empty

- Unify $\{(f(z) = f(z))\}$
 - $\{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\} = ?$

11/2/19

64

Example

- x, y, z variables, f, g constructors
- Pick a pair: $(f(z) = f(z))$

- Unify $\{(f(z) = f(z))\}$
 - $\{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\} = ?$

11/2/19

65

Example

- x, y, z variables, f, g constructors
- Pick a pair: $(f(z) = f(z))$
- Delete
- Unify $\{(f(z) = f(z))\}$
 - $\{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\} =$Unify $\{\} \circ \{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\}$

11/2/19

66

Example

- x, y, z variables, f, g constructors
- Unify $\{\} \circ \{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\} = ?$

11/2/19

67

Example

- x, y, z variables, f, g constructors
- $\{\}$ is empty
- Unify $\{\} =$ identity function
- Unify $\{\} \circ \{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\} = \{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\}$

11/2/19

68

Example

- Unify $\{(f(x) = f(g(f(z), y))), (g(y, y) = x)\} = \{y \rightarrow f(z); x \rightarrow g(f(z), f(z))\}$

$$f(x, y) = f(g(f(z), y)) \\ \rightarrow f(g(f(z), f(z))) = f(g(f(z), f(z)))$$

$$g(y, y) = x \\ \rightarrow g(f(z), f(z)) = g(f(z), f(z))$$

11/2/19

69

Example of Failure: Decompose

- Unify $\{(f(x, g(y)) = f(h(y), x))\}$
- Decompose: $(f(x, g(y)) = f(h(y), x))$
- = Unify $\{(x = h(y)), (g(y) = x)\}$
- Orient: $(g(y) = x)$
- = Unify $\{(x = h(y)), (x = g(y))\}$
- Eliminate: $(x = h(y))$
- Unify $\{(h(y) = g(y))\} \circ \{x \rightarrow h(y)\}$
- No rule to apply! Decompose fails!

11/2/19

70

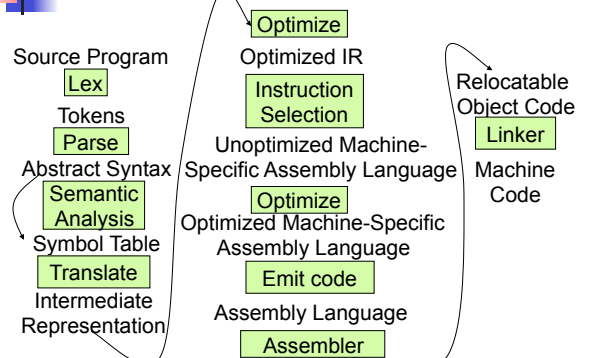
Example of Failure: Occurs Check

- Unify $\{(f(x, g(x)) = f(h(x), x))\}$
- Decompose: $(f(x, g(x)) = f(h(x), x))$
- = Unify $\{(x = h(x)), (g(x) = x)\}$
- Orient: $(g(x) = x)$
- = Unify $\{(x = h(x)), (x = g(x))\}$
- No rules apply.

11/2/19

71

Major Phases of a Compiler



Modified from "Modern Compiler Implementation in ML", by Andrew Appel

Meta-discourse

- Language Syntax and Semantics
- Syntax
 - Regular Expressions, DFSAs and NDFSAs
 - Grammars
- Semantics
 - Natural Semantics
 - Transition Semantics

11/2/19

73

Language Syntax

- Syntax is the description of which strings of symbols are meaningful expressions in a language
- It takes more than syntax to understand a language; need meaning (semantics) too
- Syntax is the entry point

11/2/19

74

Syntax of English Language

- Pattern 1

Subject	Verb
David	sings
The dog	barked
Susan	yawned

- Pattern 2

Subject	Verb	Direct Object
David	sings	ballads
The professor	wants	to retire
The jury	found	the defendant guilty

11/2/19

75

Elements of Syntax

- Character set – previously always ASCII, now often 64 character sets
- Keywords – usually reserved
- Special constants – cannot be assigned to
- Identifiers – can be assigned to
- Operator symbols
- Delimiters (parenthesis, braces, brackets)
- Blanks (aka white space)

11/2/19

76

Elements of Syntax

- Expressions
 - if ... then begin ... ; ... end else begin ... ; ... end
- Type expressions
 - $typexpr_1 \rightarrow typexpr_2$
- Declarations (in functional languages)
 - let $pattern_1 = expr_1$ in $expr$
- Statements (in imperative languages)
 - $a = b + c$
- Subprograms
 - let $pattern_1 =$ let rec inner = ... in $expr$

11/2/19

77

Elements of Syntax

- Modules
- Interfaces
- Classes (for object-oriented languages)

11/2/19

78



Lexing and Parsing

- Converting strings to abstract syntax trees done in two phases
 - **Lexing:** Converting string (or streams of characters) into lists (or streams) of tokens (the “words” of the language)
 - Specification Technique: Regular Expressions
 - **Parsing:** Convert a list of tokens into an abstract syntax tree
 - Specification Technique: BNF Grammars

11/2/19

79



Formal Language Descriptions

- Regular expressions, regular grammars, finite state automata
- Context-free grammars, BNF grammars, syntax diagrams
- Whole family more of grammars and automata – covered in automata theory

11/2/19

80



Grammars

- Grammars are formal descriptions of which strings over a given character set are in a particular language
- Language designers write grammar
- Language implementers use grammar to know what programs to accept
- Language users use grammar to know how to write legitimate programs

11/2/19

81