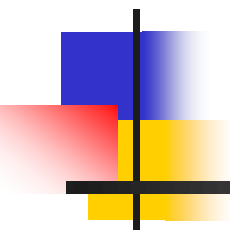


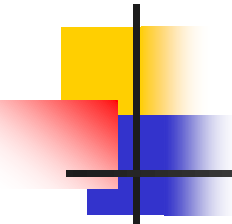
Programming Languages and Compilers (CS 421)



William Mansky

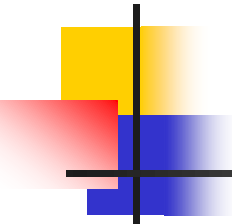
<http://courses.engr.illinois.edu/cs421/>

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



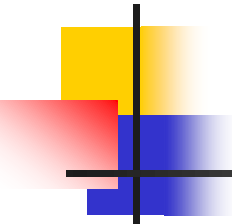
Why Data Types?

- Data types play a key role in:
 - *Data abstraction* in the design of programs
 - *Type checking* in the analysis of programs
 - *Compile-time code generation* in the translation and execution of programs



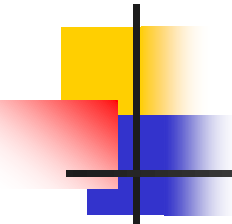
Terminology

- Type: A type t defines a set of possible data values
 - E.g. **short** in C is $\{x \mid 2^{15} - 1 \geq x \geq -2^{15}\}$
 - A value in this set is said to have type t
- Type system: rules of a language assigning types to expressions



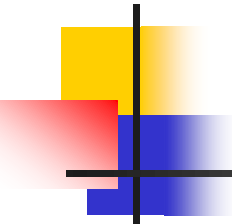
Types as Specifications

- Types describe properties
- Different type systems describe different properties, e.g.:
 - Data is read-write versus read-only
 - Operation has authority to access data
 - Data came from “right” source
 - Operation might or could not raise an exception
- Common type systems focus on types describing same data layout and access methods



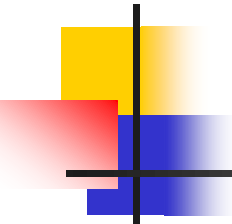
Sound Type System

- If an expression is assigned type t , and it evaluates to a value v , then v is in the set of values defined by t
- SML, OCaml, Scheme and Ada have sound type systems
- Most implementations of C and C++ do not



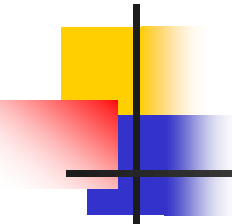
Strongly Typed Language

- When no application of an operator to arguments can lead to a run-time type error, language is *strongly typed*
 - E.g.: `1 + 2.3;;`
- Depends on definition of “type error”
 - Is an array bounds error a type error?



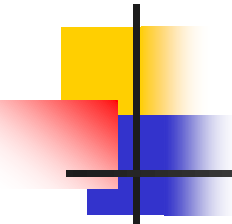
Type Checking

- When is $\text{op}(\text{arg1}, \dots, \text{argn})$ allowed?
- *Type checking* assures that operations are applied to the right number of arguments of the right types
 - May include implicit coercion
- Used to resolve overloaded operations and catch type errors



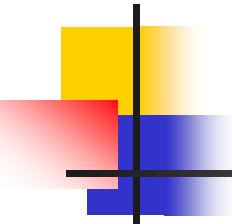
Static Type Checking

- Performed after parsing, before code generation
- Type of every variable and signature of every operator must be known at compile time
- Catches many programming errors at earliest point
- Can't check types that depend on dynamically computed values
 - E.g. array bounds



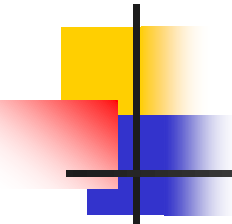
Static Type Checking

- Typically places restrictions on languages
 - Garbage collection
 - References instead of pointers
 - All variables initialized when created
 - Variable only used at one type



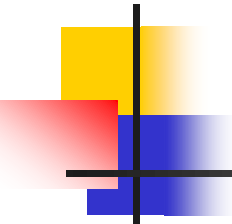
Type Declarations

- *Type declarations*: explicit assignment of types to variables (signatures to functions) in the code of a program
 - Must be checked in a strongly typed language
 - May not be necessary for strong typing or even static



Type Inference

- *Type inference*: A program analysis to assign a type to an expression from the program context of the expression
 - Fully static type inference first introduced by Robin Milner in ML
 - Haskell, OCaml, SML use type inference

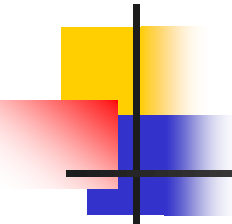


Format of Type Judgments

- A *type judgment* has the form

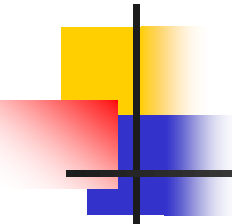
$$\Gamma \vdash \text{exp} : \tau$$

- Γ is a typing environment
 - Supplies the types of variables and functions
 - Γ is a list of the form $[x : \sigma, \dots]$
- exp is a program expression
- τ is a type to be assigned to exp
- $\vdash$ pronounced “turnstile” or “entails”



Example Valid Type Judgments

- $[] \vdash \text{true or false} : \text{bool}$
- $[x : \text{int}] \vdash x + 3 : \text{int}$
- $[p : \text{int} \rightarrow \text{string}] \vdash p\ 5 : \text{string}$



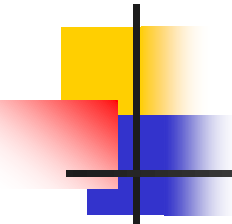
Format of Typing Rules

Assumptions

$$\frac{\Gamma_1 \vdash \text{exp}_1 : \tau_1 \quad \dots \quad \Gamma_n \vdash \text{exp}_n : \tau_n}{\Gamma \vdash \text{exp} : \tau}$$

Conclusion

- Idea: Type of expression determined by type of components
- Rule without assumptions is called an *axiom*
- Γ may be omitted when empty



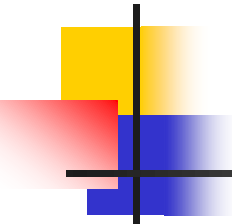
Format of Typing Rules

Assumptions

$$\frac{\Gamma_1 \vdash \text{exp}_1 : \tau_1 \quad \dots \quad \Gamma_n \vdash \text{exp}_n : \tau_n}{\Gamma \vdash \text{exp} : \tau}$$

Conclusion

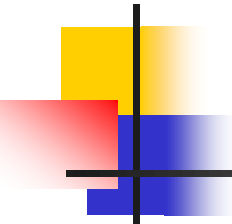
- Γ, exp, τ are *parameterized* environments, expressions and types - *i.e.* may contain *meta-variables*



Axioms - Constants

$$\frac{}{\vdash n : \text{int}} \quad (\text{assuming } n \text{ is an integer constant})$$
$$\frac{}{\vdash \text{true} : \text{bool}}$$
$$\frac{}{\vdash \text{false} : \text{bool}}$$

- These rules hold under any typing environment
- n is a meta-variable

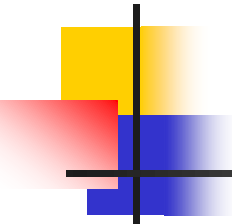


Axioms - Variables

Notation: Let $\Gamma(x) = \sigma$ if $x : \sigma \in \Gamma$ and there is no $x : \tau$ to the left of $x : \sigma$ in Γ

Variable axiom:

$$\frac{}{\Gamma \vdash x : \sigma} \quad \text{if } \Gamma(x) = \sigma$$



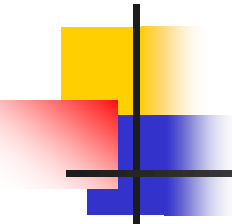
Simple Rules - Arithmetic

Primitive operators ($\oplus \in \{ +, -, *, \dots \}$):

$$\frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash e_1 \oplus e_2 : \text{int}}$$

Relations ($\sim \in \{ <, >, =, <=, >= \}$):

$$\frac{\Gamma \vdash e_1 : \text{int} \quad \Gamma \vdash e_2 : \text{int}}{\Gamma \vdash e_1 \sim e_2 : \text{bool}}$$

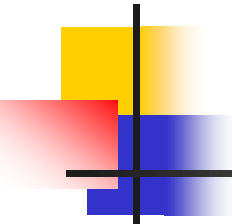


Simple Rules - Booleans

Connectives

$$\frac{\Gamma \vdash e_1 : \text{bool} \quad \Gamma \vdash e_2 : \text{bool}}{\Gamma \vdash e_1 \ \&\& \ e_2 : \text{bool}}$$

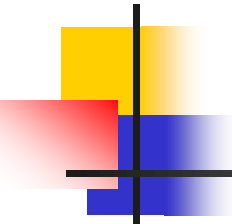
$$\frac{\Gamma \vdash e_1 : \text{bool} \quad \Gamma \vdash e_2 : \text{bool}}{\Gamma \vdash e_1 \ || \ e_2 : \text{bool}}$$



Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Start building the proof tree from the bottom up

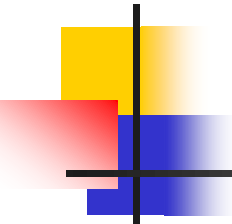
$$\frac{\quad}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}} ?$$



Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Which rule has this as a conclusion?

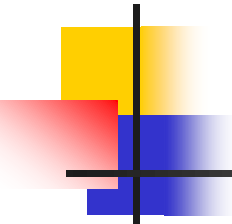
$$\frac{?}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$



Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Booleans: $\parallel$

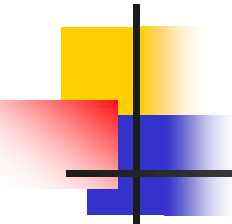
$$\frac{\Gamma \vdash y : \text{bool} \quad \Gamma \vdash x + 3 > 6 : \text{bool}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$



Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Pick an assumption to prove

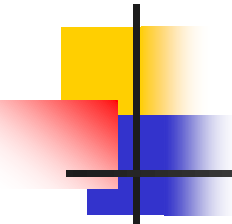
$$\frac{\frac{?}{\Gamma \vdash y : \text{bool}} \quad \Gamma \vdash x + 3 > 6 : \text{bool}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$



Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Which rule has this as a conclusion?

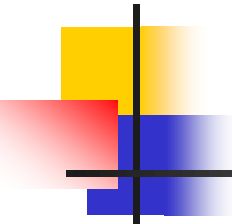
$$\frac{\frac{?}{\Gamma \vdash y : \text{bool}} \quad \Gamma \vdash x + 3 > 6 : \text{bool}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$



Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Axiom for variables

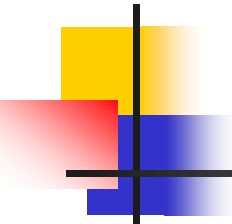
$$\frac{\Gamma \vdash y : \text{bool} \quad \Gamma \vdash x + 3 > 6 : \text{bool}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$



Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Pick an assumption to prove

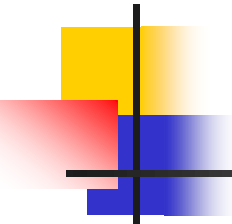
$$\frac{\frac{}{\Gamma \vdash y : \text{bool}} \quad \frac{?}{\Gamma \vdash x + 3 > 6 : \text{bool}}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$



Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Which rule has this as a conclusion?

$$\frac{\frac{}{\Gamma \vdash y : \text{bool}} \quad \frac{\frac{}{\Gamma \vdash x + 3 > 6 : \text{bool}}{?}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$



Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Arithmetic relations

$$\frac{\frac{}{\Gamma \vdash y : \text{bool}} \quad \frac{\Gamma \vdash x + 3 : \text{int} \quad \Gamma \vdash 6 : \text{int}}{\Gamma \vdash x + 3 > 6 : \text{bool}}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$

Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Pick an assumption to prove

$$\frac{\frac{\Gamma \vdash y : \text{bool}}{\Gamma \vdash y : \text{bool}} \quad \frac{\Gamma \vdash x + 3 : \text{int} \quad \frac{\Gamma \vdash 6 : \text{int}}{?}}{\Gamma \vdash x + 3 > 6 : \text{bool}}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$

Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Which rule has this as a conclusion?

$$\frac{\frac{\Gamma \vdash y : \text{bool}}{\Gamma \vdash y : \text{bool}} \quad \frac{\Gamma \vdash x + 3 : \text{int} \quad \frac{\Gamma \vdash 6 : \text{int}}{?}}{\Gamma \vdash x + 3 > 6 : \text{bool}}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$

Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Axiom for constants

$$\frac{\frac{}{\Gamma \vdash y : \text{bool}} \quad \frac{\frac{\Gamma \vdash x + 3 : \text{int} \quad \overline{\Gamma \vdash 6 : \text{int}}}{\Gamma \vdash x + 3 > 6 : \text{bool}}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}}$$

Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Pick an assumption to prove

$$\frac{\frac{\Gamma \vdash y : \text{bool}}{\Gamma \vdash y : \text{bool}} \quad \frac{\frac{\frac{\Gamma \vdash x + 3 : \text{int}}{\Gamma \vdash x + 3 : \text{int}} \quad \frac{\Gamma \vdash 6 : \text{int}}{\Gamma \vdash 6 : \text{int}}}{\Gamma \vdash x + 3 > 6 : \text{bool}}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$

Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Which rule has this as a conclusion?

$$\frac{\frac{\Gamma \vdash y : \text{bool}}{\Gamma \vdash y : \text{bool}} \quad \frac{\frac{\frac{\Gamma \vdash x + 3 : \text{int}}{\Gamma \vdash x + 3 : \text{int}} \quad \frac{\Gamma \vdash 6 : \text{int}}{\Gamma \vdash 6 : \text{int}}}{\Gamma \vdash x + 3 > 6 : \text{bool}}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$

Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Arithmetic operations

$$\frac{\frac{\Gamma \vdash x : \text{int} \quad \Gamma \vdash 3 : \text{int}}{\Gamma \vdash x + 3 : \text{int}} \quad \frac{\Gamma \vdash 6 : \text{int}}{\Gamma \vdash x + 3 > 6 : \text{bool}}}{\frac{\Gamma \vdash y : \text{bool} \quad \Gamma \vdash x + 3 > 6 : \text{bool}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}}$$

Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Pick an assumption to prove

$$\frac{\frac{\frac{\Gamma \vdash x : \text{int}}{\Gamma \vdash x : \text{int}} \quad \frac{\frac{\Gamma \vdash 3 : \text{int}}{\Gamma \vdash 3 : \text{int}}}{\Gamma \vdash x + 3 : \text{int}}} \quad \frac{\Gamma \vdash y : \text{bool}}{\Gamma \vdash y : \text{bool}} \quad \frac{\Gamma \vdash 6 : \text{int}}{\Gamma \vdash 6 : \text{int}}}{\Gamma \vdash x + 3 > 6 : \text{bool}} \quad \frac{\Gamma \vdash y : \text{bool} \quad \Gamma \vdash x + 3 > 6 : \text{bool}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$

Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Pick an assumption to prove

$$\frac{\frac{\frac{\Gamma \vdash x : \text{int}}{\Gamma \vdash x : \text{int}} \quad \frac{\frac{\Gamma \vdash 3 : \text{int}}{\Gamma \vdash 3 : \text{int}}}{\Gamma \vdash x + 3 : \text{int}}} \quad \frac{\Gamma \vdash y : \text{bool}}{\Gamma \vdash y : \text{bool}} \quad \frac{\Gamma \vdash 6 : \text{int}}{\Gamma \vdash 6 : \text{int}}}{\Gamma \vdash x + 3 > 6 : \text{bool}} \quad \frac{\Gamma \vdash y : \text{bool} \quad \Gamma \vdash x + 3 > 6 : \text{bool}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$

Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Axiom for constants

$$\frac{\frac{\Gamma \vdash x : \text{int} \quad \overline{\Gamma \vdash 3 : \text{int}}}{\Gamma \vdash x + 3 : \text{int}} \quad \overline{\Gamma \vdash 6 : \text{int}}}{\frac{\Gamma \vdash y : \text{bool} \quad \Gamma \vdash x + 3 > 6 : \text{bool}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}}$$

Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Pick an assumption to prove

$$\frac{\frac{\frac{\Gamma \vdash x : \text{int}}{\Gamma \vdash x + 3 : \text{int}} \quad \frac{\Gamma \vdash 3 : \text{int}}{\Gamma \vdash 6 : \text{int}}}{\Gamma \vdash x + 3 > 6 : \text{bool}} \quad \Gamma \vdash y : \text{bool}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$

Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- Axiom for variables

$$\frac{\frac{\frac{}{\Gamma \vdash x : \text{int}} \quad \frac{}{\Gamma \vdash 3 : \text{int}}}{\Gamma \vdash x + 3 : \text{int}} \quad \frac{}{\Gamma \vdash 6 : \text{int}}}{\Gamma \vdash x + 3 > 6 : \text{bool}} \quad \frac{}{\Gamma \vdash y : \text{bool}}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$

Simple Example

- Let $\Gamma = [x:\text{int} ; y:\text{bool}]$
- Show $\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}$
- No more assumptions! QED!

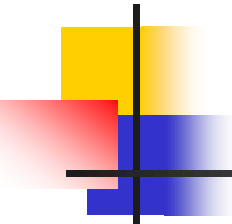
$$\frac{\frac{\frac{}{\Gamma \vdash x : \text{int}} \quad \frac{}{\Gamma \vdash 3 : \text{int}}}{\Gamma \vdash x + 3 : \text{int}} \quad \frac{}{\Gamma \vdash 6 : \text{int}}}{\Gamma \vdash x + 3 > 6 : \text{bool}} \quad \frac{}{\Gamma \vdash y : \text{bool}}}{\Gamma \vdash y \parallel (x + 3 > 6) : \text{bool}}$$

Type Variables in Rules

- If_then_else rule:

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

- τ is a type variable (meta-variable)
- Can take any type at all
- All instances in a rule application must get same type
- Then branch, else branch and if_then_else must all have same type

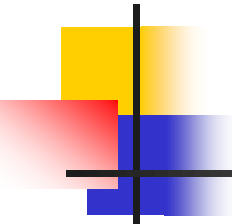


Function Application

- Application rule:

$$\frac{\Gamma \vdash e_1 : \tau_1 \rightarrow \tau_2 \quad \Gamma \vdash e_2 : \tau_1}{\Gamma \vdash (e_1 \ e_2) : \tau_2}$$

- If you have a function expression e_1 of type $\tau_1 \rightarrow \tau_2$ applied to an argument of type τ_1 , the resulting expression has type τ_2



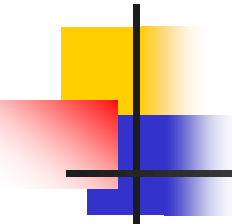
Application Examples

$$\frac{\Gamma \vdash \text{print_int} : \text{int} \rightarrow \text{unit} \quad \Gamma \vdash 5 : \text{int}}{\Gamma \vdash (\text{print_int } 5) : \text{unit}}$$

- $e_1 = \text{print_int}, \quad e_2 = 5,$
- $\tau_1 = \text{int}, \quad \tau_2 = \text{unit}$

$$\frac{\Gamma \vdash \text{map print_int} : \text{int list} \rightarrow \text{unit list} \quad \Gamma \vdash [3;7] : \text{int list}}{\Gamma \vdash (\text{map print_int } [3; 7]) : \text{unit list}}$$

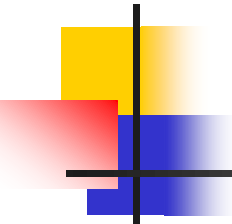
- $e_1 = \text{map print_int}, \quad e_2 = [3; 7],$
- $\tau_1 = \text{int list}, \quad \tau_2 = \text{unit list}$



Fun Rule

- Rules describe types, but also how the environment Γ may change
- Can only do what rule allows!
- fun rule:

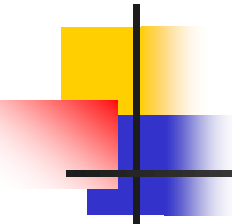
$$\frac{[x : \tau_1] + \Gamma \vdash e : \tau_2}{\Gamma \vdash \text{fun } x \rightarrow e : \tau_1 \rightarrow \tau_2}$$



Fun Examples

$$\frac{[y : \text{int}] + \Gamma \vdash y + 3 : \text{int}}{\Gamma \vdash \text{fun } y \rightarrow y + 3 : \text{int} \rightarrow \text{int}}$$

$$\frac{[f : \text{int} \rightarrow \text{bool}] + \Gamma \vdash f \ 2 :: [\text{true}] : \text{bool list}}{\Gamma \vdash (\text{fun } f \rightarrow f \ 2 :: [\text{true}]) : (\text{int} \rightarrow \text{bool}) \rightarrow \text{bool list}}$$



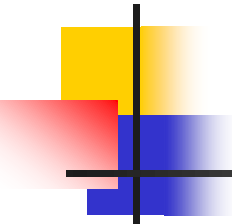
(Monomorphic) Let and Let Rec

- let rule:

$$\frac{\Gamma \vdash e_1 : \tau_1 \quad [x : \tau_1] + \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (\text{let } x = e_1 \text{ in } e_2) : \tau_2}$$

- let rec rule:

$$\frac{[x : \tau_1] + \Gamma \vdash e_1 : \tau_1 \quad [x : \tau_1] + \Gamma \vdash e_2 : \tau_2}{\Gamma \vdash (\text{let rec } x = e_1 \text{ in } e_2) : \tau_2}$$



Example

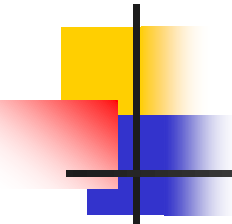
- Which rule do we apply?

?

$\vdash$ (let rec one = 1 :: one in
 let x = 2 in
 fun y -> (x :: y :: one)) : int $\rightarrow$ int list

■ Let rec rule:

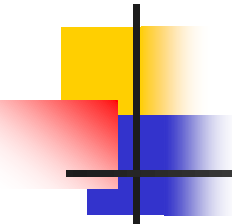
$\frac{\textcircled{1} \quad \text{[one : int list]} \vdash (1 :: \text{one}) : \text{int list}}{\vdash (\text{let rec one} = 1 :: \text{one in let x} = 2 \text{ in fun y} \rightarrow (x :: y :: \text{one})) : \text{int} \rightarrow \text{int list}}$	$\textcircled{2} \quad \text{[one : int list]} \vdash (\text{let x} = 2 \text{ in fun y} \rightarrow (x :: y :: \text{one})) : \text{int} \rightarrow \text{int list}$
--	--



Proof of 1

- Which rule?

$[one : \text{int list}] \vdash (1 :: one) : \text{int list}$



Proof of 1

■ Application

③

$[one : \text{int list}] \vdash$

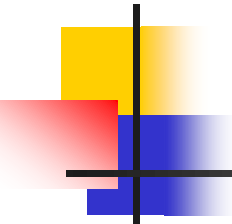
$((::) 1) : \text{int list} \rightarrow \text{int list}$

④

$[one : \text{int list}] \vdash$

$one : \text{int list}$

$[one : \text{int list}] \vdash (1 :: one) : \text{int list}$



Proof of 3

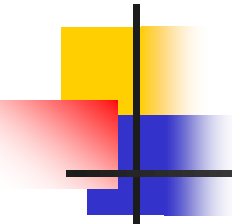
Constants Rule

$$[\text{one} : \text{int list}] \vdash$$
$$(::) : \text{int} \rightarrow \text{int list} \rightarrow \text{int list}$$

$$[\text{one} : \text{int list}] \vdash ((::) \ 1) : \text{int list} \rightarrow \text{int list}$$

Constants Rule

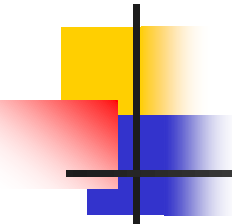
$$[\text{one} : \text{int list}] \vdash$$
$$1 : \text{int}$$



Proof of 4

- Rule for variables

$$\overline{[\text{one} : \text{int list}] \vdash \text{one} : \text{int list}}$$



Proof of 2

⑤ $[x : \text{int}; \text{one} : \text{int list}] \vdash$

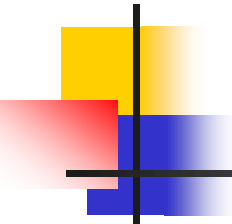
■ Constant

$\text{fun } y \rightarrow$

$(x :: y :: \text{one}))$

$[\text{one} : \text{int list}] \vdash 2:\text{int} \quad : \text{int} \rightarrow \text{int list}$

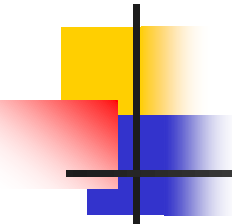
$[\text{one} : \text{int list}] \vdash (\text{let } x = 2 \text{ in}$
 $\text{fun } y \rightarrow (x :: y :: \text{one})) : \text{int} \rightarrow \text{int list}$



Proof of 5

?

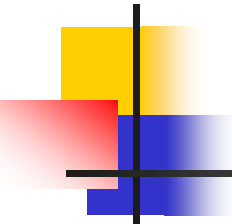
$$[x : \text{int}; \text{one} : \text{int list}] \vdash \text{fun } y \rightarrow (x :: y :: \text{one}))$$
$$: \text{int} \rightarrow \text{int list}$$



Proof of 5

?

$$\frac{[y : \text{int}; x : \text{int}; \text{one} : \text{int list}] \vdash (x :: y :: \text{one}) : \text{int list}}{[x : \text{int}; \text{one} : \text{int list}] \vdash \text{fun } y \rightarrow (x :: y :: \text{one})) : \text{int} \rightarrow \text{int list}}$$

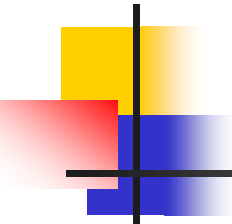


Proof of 5

⑥

$$\frac{\frac{[y:\text{int}; x:\text{int}; \text{one} : \text{int list}] \vdash ((::) x):\text{int list} \rightarrow \text{int list} \quad [y:\text{int}; x:\text{int}; \text{one} : \text{int list}] \vdash (y :: \text{one}) : \text{int list}}{[y : \text{int}; x:\text{int}; \text{one} : \text{int list}] \vdash (x :: y :: \text{one}) : \text{int list}}}{[x : \text{int}; \text{one} : \text{int list}] \vdash \text{fun } y \rightarrow (x :: y :: \text{one})) : \text{int} \rightarrow \text{int list}}$$

⑦



Proof of 6

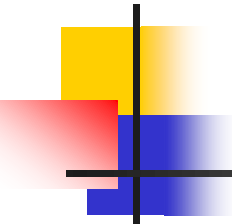
Constant

Variable

$$[\dots] \vdash (::)$$
$$: \text{int} \rightarrow \text{int list} \rightarrow \text{int list}$$

$$[\dots; x : \text{int}; \dots] \vdash x : \text{int}$$

$$[y : \text{int}; x : \text{int}; \text{one} : \text{int list}] \vdash ((::) x)$$
$$: \text{int list} \rightarrow \text{int list}$$



Proof of 7

Pf of 6 $[y/x]$

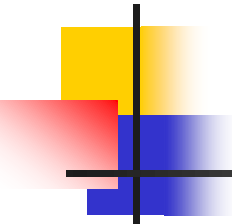
•
•
•

Variable

$$[y : \text{int}; \dots] \vdash ((::) y) : \text{int list} \rightarrow \text{int list}$$

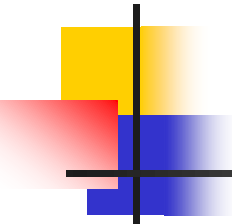
$$[\dots; \text{one} : \text{int list}] \vdash \text{one} : \text{int list}$$

$$[y : \text{int}; x : \text{int}; \text{one} : \text{int list}] \vdash (y :: \text{one}) : \text{int list}$$



Curry-Howard Isomorphism

- Type Systems are logics; logics are type systems
- Types are propositions; propositions are types
- Terms (expressions) are proofs; proofs are terms
- Functions space arrow corresponds to implication; application corresponds to modus ponens



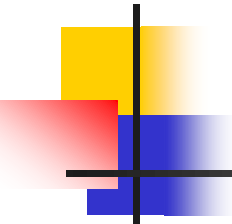
Curry - Howard Isomorphism

■ Modus Ponens

$$\frac{A \Rightarrow B \quad A}{B}$$

• Application

$$\frac{\Gamma \vdash e_1 : \alpha \rightarrow \beta \quad \Gamma \vdash e_2 : \alpha}{\Gamma \vdash (e_1 \ e_2) : \beta}$$



Remaining Problems

- The above system can't handle polymorphism as in OCAML
- No type variables in type language (only meta-variables in the logic)
- Would need:
 - Object level type variables and some kind of type quantification
 - **let** and **let rec** rules to introduce polymorphism
 - Explicit rule to eliminate (instantiate) polymorphism