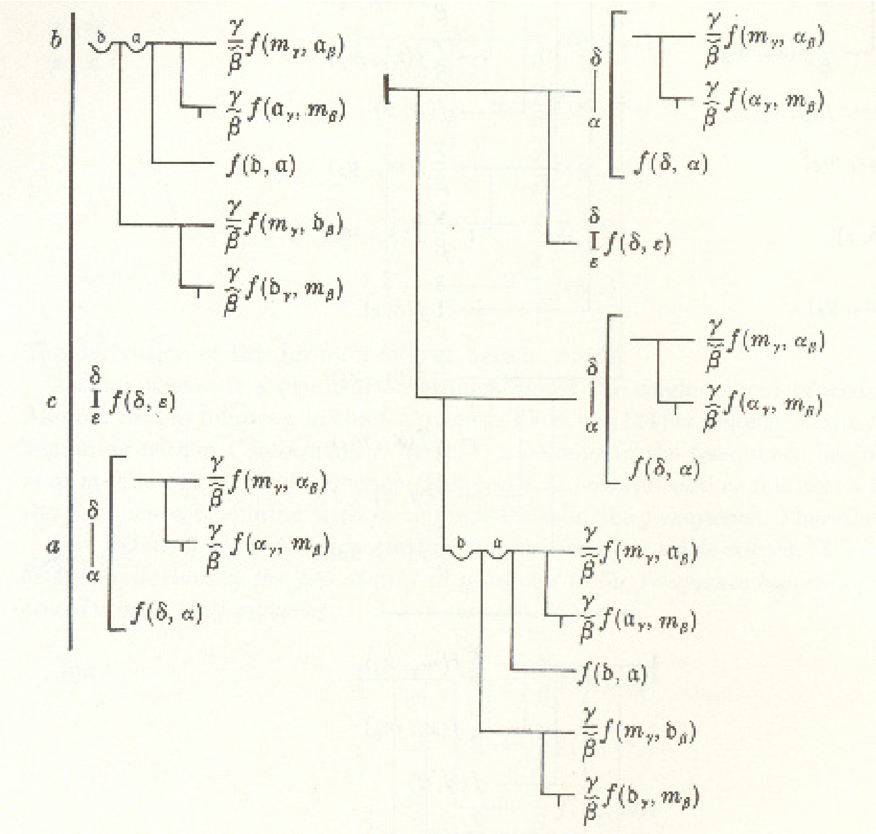
From Frege to Gosling: 19'th Century Logic and 21'st Century Programming Languages

Philip Wadler

Avaya Labs

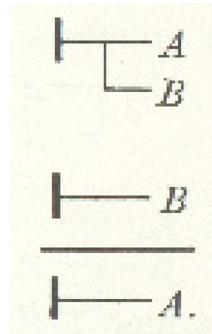
Gottlob Frege, *Begriffsschrift*, 1879



Part I

The Curry-Howard Isomorphism

Modus ponens Frege, 1879



Gentzen, 1934

$$\frac{\vdash B \rightarrow A \qquad \vdash B}{\vdash A} \rightarrow\text{-}\mathbf{I}$$

Inference rules

$$\frac{}{A_1, \ldots, A_n \vdash A_i}\ \text{Id}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \to B}\ \to\text{-I} \qquad \frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}\ \to\text{-E}$$

$$\frac{\Gamma \vdash B}{\Gamma \vdash \forall X.\, B}\ \forall\text{-I}\ (X \notin \Gamma) \qquad \frac{\Gamma \vdash \forall X.\, B}{\Gamma \vdash B[A/X]}\ \forall\text{-E}$$

Inference rules

$$\frac{}{x_1 : A_1, \ldots, x_n : A_n \vdash x_i : A_i} \text{ Id}$$

$$\frac{\Gamma, x : A \vdash u : B}{\Gamma \vdash \lambda x{:}A.\, u : A \to B} \to\text{-I} \qquad \frac{\Gamma \vdash s : A \to B \qquad \Gamma \vdash t : A}{\Gamma \vdash s(t) : B} \to\text{-E}$$

$$\frac{\Gamma \vdash u : B}{\Gamma \vdash \Lambda X.\, u : \forall X.\, B} \forall\text{-I } (X \notin \Gamma) \qquad \frac{\Gamma \vdash s : \forall X.\, B}{\Gamma \vdash s\langle A \rangle : B[A/X]} \forall\text{-E}$$

A proof

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{A \to X \vdash A \to X}\;\text{Id} \qquad \Gamma \vdash A}{\Gamma,\; A \to X \vdash X}\;\to\text{-E}}{\Gamma \vdash (A \to X) \to X}\;\to\text{-I}}{\Gamma \vdash \forall X.\,(A \to X) \to X}\;\forall\text{-I}
$$

A proof

$$
\dfrac{\dfrac{\dfrac{\dfrac{}{f : A \to X \vdash f : A \to X}\,\text{Id} \qquad \Gamma \vdash t : A}{\Gamma,\, f : A \to X \vdash f(t) : X}\,\to\text{-E}}{\Gamma \vdash \lambda f{:}A \to X.\, f(t) : (A \to X) \to X}\,\to\text{-I}}{\Gamma \vdash \Lambda X.\, \lambda f{:}A \to X.\, f(t) : \forall X.\, (A \to X) \to X}\,\forall\text{-I}
$$

Another proof

$$
\dfrac{\dfrac{\Gamma \vdash \forall X.\,(A \to X) \to X}{\Gamma \vdash (A \to A) \to A}\ \forall\text{-E} \qquad \dfrac{\dfrac{}{A \vdash A}\ \text{Id}}{\vdash A \to A}\ {\to}\text{-I}}{\Gamma \vdash A}\ {\to}\text{-E}
$$

Another proof

$$
\dfrac{
  \dfrac{\Gamma \vdash s : \forall X.\,(A \to X) \to X}{\Gamma \vdash s\langle A\rangle : (A \to A) \to A}\;\forall\text{-E}
  \qquad
  \dfrac{\dfrac{}{x : A \vdash x : A}\;\text{Id}}{\vdash \lambda x{:}A.\,x : A \to A}\;\to\text{-I}
}{\Gamma \vdash s\langle A\rangle(\lambda x{:}A.\,x) : A}\;\to\text{-E}
$$

A combined proof

$$\dfrac{}{A \to X \vdash A \to X}\ \text{Id}$$

$$\dfrac{A \to X \vdash A \to X \qquad \Gamma \vdash A}{\Gamma, A \to X \vdash X}\ {\to}\text{-E}$$

$$\dfrac{\Gamma, A \to X \vdash X}{\Gamma \vdash (A \to X) \to X}\ {\to}\text{-I}$$

$$\dfrac{\Gamma \vdash (A \to X) \to X}{\Gamma \vdash \forall X.\,(A \to X) \to X}\ \forall\text{-I}$$

$$\dfrac{\Gamma \vdash \forall X.\,(A \to X) \to X}{\Gamma \vdash (A \to A) \to A}\ \forall\text{-E}$$

$$\dfrac{}{A \vdash A}\ \text{Id}$$

$$\dfrac{A \vdash A}{\vdash A \to A}\ {\to}\text{-I}$$

$$\dfrac{\Gamma \vdash (A \to A) \to A \qquad \vdash A \to A}{\Gamma \vdash A}\ {\to}\text{-E}$$

A combined proof

$$
\dfrac{
\dfrac{
\dfrac{
\dfrac{
\dfrac{}{f : A \to X \vdash f : A \to X}\,\text{Id} \qquad \Gamma \vdash t : A
}{
\Gamma, f : A \to X \vdash f(t) : X
}\,\to\text{-E}
}{
\Gamma \vdash \lambda f{:}A \to X.\, f(t) : (A \to X) \to X
}\,\to\text{-I}
}{
\Gamma \vdash \Lambda X.\, \lambda f{:}A \to X.\, f(t) : \forall X.\, (A \to X) \to X
}\,\forall\text{-I}
}{
\Gamma \vdash (\Lambda X.\, \lambda f{:}A \to X.\, f(t))\langle A \rangle : (A \to A) \to A
}\,\forall\text{-E}
\qquad
\dfrac{
\dfrac{}{x : A \vdash x : A}\,\text{Id}
}{
\vdash \lambda x{:}A.\, x : A \to A
}\,\to\text{-I}
}{
\Gamma \vdash (\Lambda X.\, \lambda f{:}A \to X.\, f(t))\langle A \rangle(\lambda x{:}A.\, x) : A
}\,\to\text{-E}
$$

Reductions

$$\dfrac{\dfrac{\Gamma, \quad A \vdash \quad B}{\Gamma \vdash \quad A \to B}\ \to\text{-I} \qquad \Gamma \vdash \quad A}{\Gamma \vdash \quad B}\ \to\text{-E} \quad \Rightarrow \quad \Gamma \vdash \quad B$$

$$\dfrac{\dfrac{\Gamma \vdash \quad B}{\Gamma \vdash \quad \forall X.\, B}\ \forall\text{-I}}{\Gamma \vdash \quad B[A/X]}\ \forall\text{-E} \quad \Rightarrow \quad \Gamma \vdash \quad B[A/X]$$

Reductions

$$\dfrac{\dfrac{\Gamma, x : A \vdash u : B}{\Gamma \vdash \lambda x{:}A.\, u : A \to B}\ \to\text{-I} \qquad \Gamma \vdash t : A}{\Gamma \vdash (\lambda x{:}A.\, u)(t) : B}\ \to\text{-E} \quad \Rightarrow \quad \Gamma \vdash u[t/x] : B$$

$$\dfrac{\dfrac{\Gamma \vdash u : B}{\Gamma \vdash \Lambda X.\, u : \forall X.\, B}\ \forall\text{-I}}{\Gamma \vdash (\Lambda X.\, u)\langle A \rangle : B[A/X]}\ \forall\text{-E} \quad \Rightarrow \quad \Gamma \vdash u[A/X] : B[A/X]$$

Simplifying a proof

$$
\dfrac{
  \dfrac{
    \dfrac{
      \dfrac{
        \dfrac{\dfrac{}{f : A \to X \vdash f : A \to X}\ \text{Id} \qquad \Gamma \vdash t : A}{\Gamma,\ f : A \to X \vdash f(t) : X}\ \to\text{-E}
      }{\Gamma \vdash \lambda f{:}A \to X.\ f(t) : (A \to X) \to X}\ \to\text{-I}
    }{\Gamma \vdash \Lambda X.\ \lambda f{:}A \to X.\ f(t) : \forall X.\ (A \to X) \to X}\ \forall\text{-I}
  }{\Gamma \vdash (\Lambda X.\ \lambda f{:}A \to X.\ f(t))\langle A \rangle : (A \to A) \to A}\ \forall\text{-E}
  \qquad
  \dfrac{\dfrac{}{x : A \vdash x : A}\ \text{Id}}{\vdash \lambda x{:}A.\ x : A \to A}\ \to\text{-I}
}{\Gamma \vdash (\Lambda X.\ \lambda f{:}A \to X.\ f(t))\langle A \rangle(\lambda x{:}A.\ x) : A}\ \to\text{-E}
$$

Simplifying a proof

$$\dfrac{\dfrac{\dfrac{\dfrac{}{f : A \to A \vdash f : A \to A}\,\text{Id} \qquad \Gamma \vdash t : A}{\Gamma, f : A \to A \vdash f(t) : A}\,\to\text{-E}}{\Gamma \vdash \lambda f{:}A \to X.\, f(t) : (A \to A) \to A}\,\to\text{-I} \qquad \dfrac{\dfrac{}{x : A \vdash x : A}\,\text{Id}}{\vdash \lambda x{:}A.\, x : A \to A}\,\to\text{-I}}{\Gamma \vdash (\lambda f{:}A \to A.\, f(t))(\lambda x{:}A.\, x) : A}\,\to\text{-E}$$

Simplifying a proof

$$\dfrac{\dfrac{\dfrac{}{x : A \vdash x : A}\ \text{Id}}{\vdash \lambda x{:}A.\,x : A \to A}\ \to\text{-I} \qquad \Gamma \vdash t : A}{\Gamma \vdash (\lambda x{:}A.\,x)(t) : A}\ \to\text{-E}$$

Simplifying a proof

$$\Gamma \vdash t : A$$

Additional reductions

$$\dfrac{\dfrac{\Gamma \vdash s : A \to B \qquad \dfrac{}{x : A \vdash x : A}\,\text{Id}}{\Gamma, x : A \vdash s(x) : B}\,\to\text{-E}}{\Gamma \vdash \lambda x{:}A.\, s(x) : A \to B}\,\to\text{-I} \quad \Rightarrow \quad \Gamma \vdash s : A \to B$$

$$\dfrac{\dfrac{\Gamma \vdash s : \forall X.\, B}{\Gamma \vdash s\langle X\rangle : B}\,\forall\text{-E}}{\Gamma \vdash \Lambda X.\, s\langle X\rangle : \forall X.\, B}\,\forall\text{-I} \quad \Rightarrow \quad \Gamma \vdash s : \forall X.\, B$$

Summary

- Gottlob Frege, 1879
  logic

- Alonzo Church, 1932, 1940
  lambda calculus

- Gerhard Gentzen, 1935
  natural deduction, proof normalization

- Haskell Curry, 1958
  combinators correspond to logic

- Dag Prawitz, 1965
  proof normalization for natural deduction

- W. A. Howard, 1980
  lambda calculus corresponds to logic

Part II

Parametricity

Lists and map

$$\texttt{map} : \forall X. \forall Y. (X \to Y) \to (\texttt{list}\langle X\rangle \to \texttt{list}\langle Y\rangle)$$

For example, if $\texttt{num} : \texttt{Int} \to \texttt{Str}$ then

$$\texttt{map}\langle\texttt{Int}\rangle\langle\texttt{Str}\rangle(\texttt{num})([1, 2, 3]) = [\texttt{"i"}, \texttt{"ii"}, \texttt{"iii"}]$$

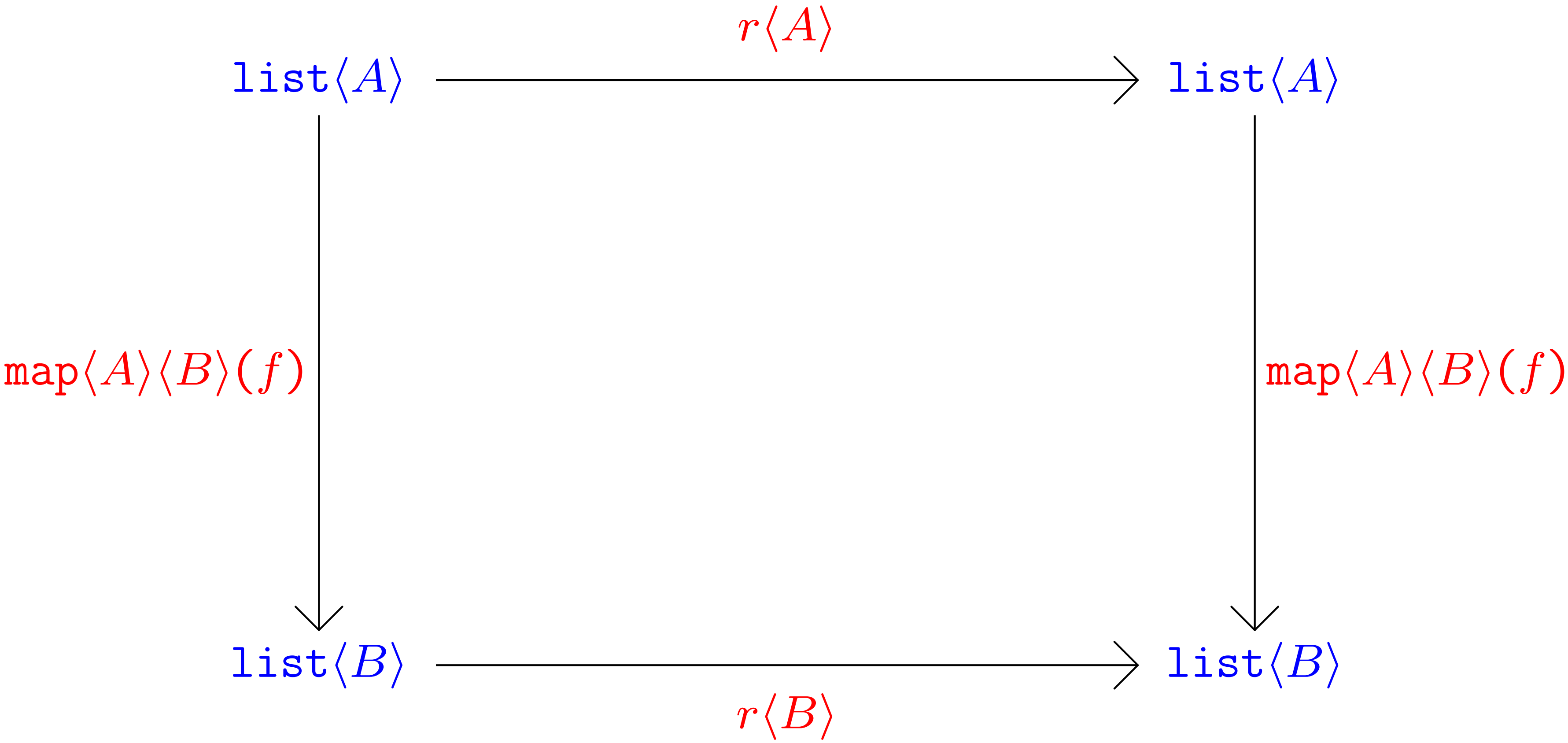
A magic trick

Think of a function with this type:

$$r : \forall X. \mathtt{list}\langle X \rangle \to \mathtt{list}\langle X \rangle$$

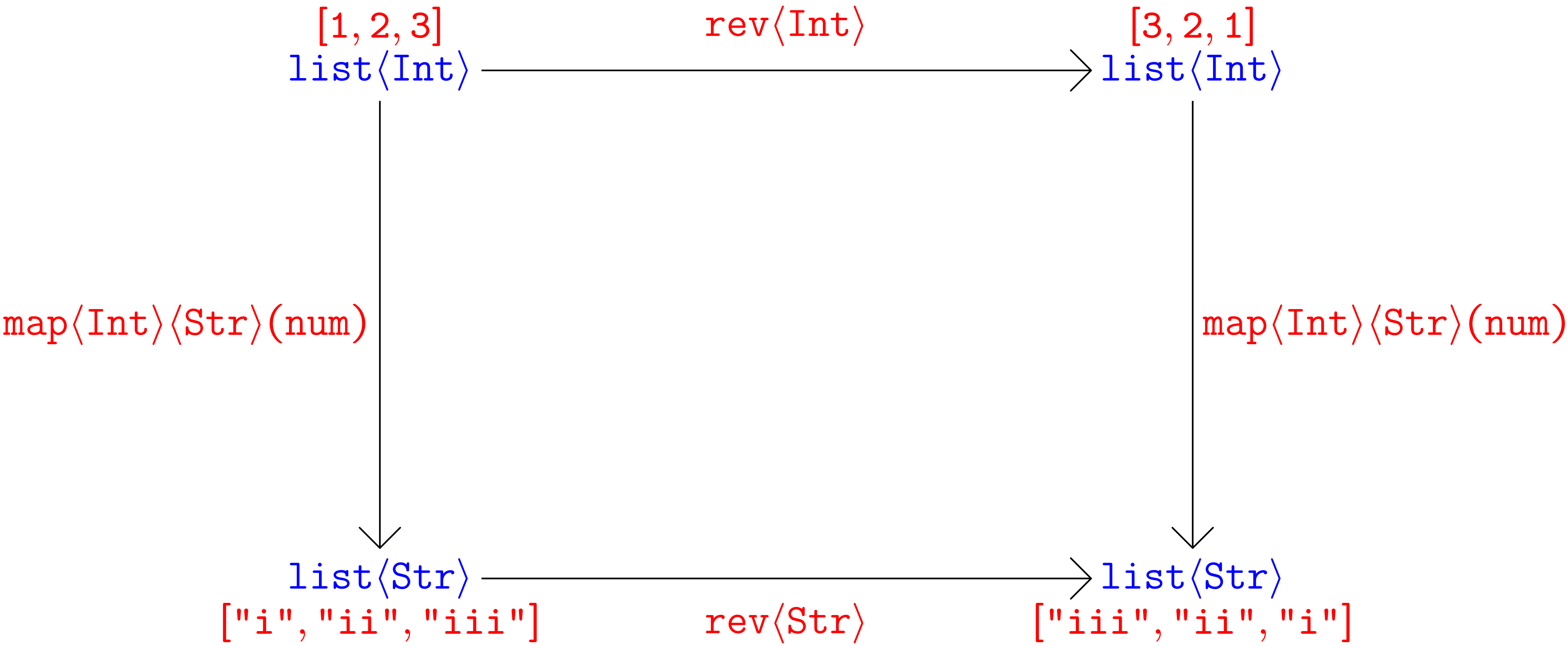
Theorems for Free!

for all $r : \forall X.\,\texttt{list}\langle X\rangle \to \texttt{list}\langle X\rangle$,
for all A, B, $f : A \to B$,



Theorems for Free! — an example

$$\text{take } r = \texttt{rev} : \forall X.\, \texttt{list}\langle X\rangle \to \texttt{list}\langle X\rangle,$$
$$\text{take } A = \texttt{Int}, B = \texttt{Str}, f = \texttt{num} : \texttt{Int} \to \texttt{Str},$$



Part III

Generic Java

Gilad Bracha, JavaSoft, Sun Microsystems

Martin Odersky, University of Lausanne

David Stoutamire, JavaSoft, Sun Microsystems

Philip Wadler, Avaya Labs

Lists in Java and GJ

	in Java	in GJ
integer list		
string list		
string list list		

Lists in Java and GJ

	in Java	in GJ
integer list	`List`	
string list		
string list list		

Lists in Java and GJ

	in Java	in GJ
integer list	`List`	
string list	`List`	
string list list		

Lists in Java and GJ

	in Java	in GJ
integer list	`List`	
string list	`List`	
string list list	`List`	

Lists in Java and GJ

	in Java	in GJ
integer list	`List`	`List<Integer>`
string list	`List`	`List<String>`
string list list	`List`	`List<List<String>>`

Support for reuse at multiple types

- C++: templates
- Ada: generics
- Standard ML, Haskell: parametric polymorphism
- Java: ???

Lists in Java

```
interface List {
  public void add (Object x);
  public Iterator iterator ();
}
interface Iterator {
  public Object next ();
  public boolean hasNext ();
}
```

Lists in GJ

```
interface List<A> {
  public void add (A x);
  public Iterator<A> iterator ();
}
interface Iterator<A> {
  public A next ();
  public boolean hasNext ();
}
```

Translating lists from GJ to Java

```
interface List {
  public void add (Object x);
  public Iterator iterator ();
}
interface Iterator {
  public Object next ();
  public boolean hasNext ();
}
```

Accessing lists in Java

```
// integer list
List xs = new LinkedList();  xs.add(new Integer(2));
Integer x = (Integer)xs.iterator().next();

// string list list
List ys = new LinkedList();  ys.add("ii");
List yss = new LinkedList();  yss.add(ys);
String y = (String)((List)yss.iterator().next()).iterator().next();

// string list treated as integer list
Integer z = (Integer)ys.iterator().next();  // run-time exception
```

Accessing lists in GJ

```
// integer list
List<Integer> xs = new LinkedList<Integer>();  xs.add(new Integer(2));
Integer x = xs.iterator().next();

// string list list
List<String> ys = new LinkedList<String>();  ys.add("ii");
List<List<String>> yss = new LinkedList<List<String>>();  yss.add(ys);
String y = yss.iterator().next().iterator().next();

// string list treated as integer list
Integer z = ys.iterator().next();  // compile-time error
```

Implementing lists in Java

```
class LinkedList implements List {
  protected class Node {
    Object elt;  Node next;
    Node (Object e) { elt=e; next=null; }
  }
  protected Node h, t;
  public LinkedList () { h=new Node(null); t=h; }
  public void add (Object elt) { t.next=new Node(elt); t=t.next; }
  public Iterator iterator () {
    return new Iterator () {
      protected Node p=h.next;
      public boolean hasNext () { return p!=null; }
      public Object next () { Object e=p.elt; p=p.next; return e; }
    };
  }
}
```

Implementing lists in GJ

```
class LinkedList<A> implements List<A> {
  protected class Node {
    A elt;  Node next;
    Node (A e) { elt=e; next=null; }
  }
  protected Node h, t;
  public LinkedList () { h=new Node(null); t=h; }
  public void add (A elt) { t.next=new Node(elt); t=t.next; }
  public Iterator<A> iterator () {
    return new Iterator<A> () {
      protected Node p=h.next;
      public boolean hasNext () { return p!=null; }
      public A next () { A e=p.elt; p=p.next; return e; }
    };
  }
}
```

Bounds: Maximum in Java

```
interface Comparable { public int compareTo (Object o); }
class Lists {
  public static Comparable max (List xs) {
    Iterator xi = xs.iterator();
    Comparable w = (Comparable)xi.next();
    while (xi.hasNext()) {
      Comparable x = (Comparable)xi.next();
      if (w.compareTo(x) < 0) w = x;
    }
    return w;
  }
}
List xs = new LinkedList(); xs.add(new Byte(0));
Byte x = (Byte)Lists.max(xs);
List ys = new LinkedList(); ys.add(new Boolean(false));
Boolean y = (Boolean)Lists.max(ys);  // run-time exception
```

Bounds: Maximum in GJ

```
interface Comparable<A> { public int compareTo (A o); }
class Lists {
  public static <A implements Comparable<A>> A max (List<A> xs) {
    Iterator<A> xi = xs.iterator();
    A w = xi.next();
    while (xi.hasNext()) {
      A x = xi.next();
      if (w.compareTo(x) < 0) w = x;
    }
    return w;
  }
}
List<Byte> xs = new LinkedList<Byte>(); xs.add(new Byte(0));
Byte x = Lists.max(xs);
List<Boolean> ys = new LinkedList<Boolean>(); ys.add(new Boolean(false));
Boolean y = Lists.max(ys);  // compile-time error
```

Arrays

Arrays and Subtyping

Covariant subtyping

```
Integer extends Object

Integer[] extends Object[]
```

A good consequence

```
class Arrays {
  void sort (Object[] oa) { ... }
}
Integer[] ia = new Integer[n];
sort(ia)  // simulates polymorphism
```

A bad consequence

```
Integer[] ia = new Integer[n];
Object[] oa = ia;
oa[0] = "hello";  // run-time exception
```

Arrays in Java

```
interface List { ...
  public Object[] toArray();
}
class LinkedList implements List { ...
  public Object[] toArray() {
    Object[] xa = new Object[this.size()];
    ... // copy list into xa
    return xa;
  }
}
List ys = new LinkedList();  ys.add("zero");
String[] ya = (String[])ys.toArray();  // run-time exception
```

Arrays in GJ — the wrong way

```
interface List<A> { ...
  public A[] toArray();
}
class LinkedList<A> implements List<A> { ...
  public A[] toArray() {
    A[] xa = new A[this.size()];  // compile-time unchecked warning
    ... // copy list into xa
    return xa;
  }
}
List<String> ys = new LinkedList<String>();  ys.add("zero");
String[] ya = ys.toArray();  // run-time exception
```

Arrays in GJ — the right way

```
interface List<A> { ...
  public A[] toArray(A[] xa);
}
class LinkedList<A> implements List<A> { ...
  public A[] toArray(A[] xa) {
    if (xa.length < this.size())
      xa = gj.lang.Array.newInstance(xa, this.size());  // no warning
    ... // copy list into xa
    return xa;
  }
}
List<String> ys = new LinkedList<String>();  ys.add("zero");
String[] ya = ys.toArray(new String[0]);  // no exception
```

Conclusion

Generic Java

- GJ supports generic types
- GJ contains Java as a subset
- GJ compiles to the Java Virtual Machine
- GJ works with Java libraries
- GJ compiler written by Martin Odersky
  distributed by Sun as javac

Higher-order functions in Java

Lambda calculus

$\lambda x{:}\mathtt{Int}.\, x + 1 \;\; : \;\; \mathtt{Int} \;\rightarrow\; \mathtt{Int}$

ML

```
fn(x) => x+1  :  int -> int
```

GJ

```
interface Function<A,B> {
  public B apply (A x);
}
new Function<Integer,Integer>() {
  public Integer apply (Integer x) {
    return new Integer(x.intValue()+1);
  }
}
```

GJ vs. C++

	GJ (erasure)	C++ (expansion)
New GJ code works with old Java code	√	×
Requirements on types explicit (bounds)	√	×
Report errors at compile-time	√	×
Avoid code bloat	√	×
Easy to in-line operators	×	√
Allow base types as type parameters	×	√

GJ is available from:

`www.research.avaya-labs.com/~wadler/gj/`

Sun is adding generic types to Java:

`http://java.sun.com/people/gbracha/`

`generics-update.html`

Part IV

Featherweight Java

Benjamin Pierce, University of Pennsylvania

Atsushi Igarashi, University of Kyoto

Philip Wadler, Avya Labs

Featherweight Java: Syntax

$$
\begin{array}{lcl}
\texttt{L} & ::= & \texttt{class C extends C \{}\overline{\texttt{C}}\ \overline{\texttt{f}}\texttt{; K }\overline{\texttt{M}}\texttt{\}} \\
\texttt{K} & ::= & \texttt{C(}\overline{\texttt{C}}\ \overline{\texttt{f}}\texttt{)\{super(}\overline{\texttt{f}}\texttt{); this.}\overline{\texttt{f}}\texttt{=}\overline{\texttt{f}}\texttt{;\}} \\
\texttt{M} & ::= & \texttt{C m(}\overline{\texttt{C}}\ \overline{\texttt{x}}\texttt{)\{ return e; \}} \\
\texttt{e} & ::= & \texttt{x} \mid \texttt{e.f} \mid \texttt{e.m(}\overline{\texttt{e}}\texttt{)} \mid \texttt{new C(}\overline{\texttt{e}}\texttt{)} \mid \texttt{(C)e}
\end{array}
$$

Typing rules

$$\frac{}{\Gamma \vdash \mathtt{x} : \Gamma(\mathtt{x})}$$

$$\frac{\Gamma \vdash \mathtt{e}_0 : \mathtt{C}_0 \qquad \mathit{fields}(\mathtt{C}_0) = \overline{\mathtt{C}}\ \overline{\mathtt{f}}}{\Gamma \vdash \mathtt{e}_0.\mathtt{f}_i : \mathtt{C}_i}$$

$$\frac{\Gamma \vdash \mathtt{e}_0 : \mathtt{C}_0 \qquad \mathit{mtype}(\mathtt{m}, \mathtt{C}_0) = \overline{\mathtt{D}} \longrightarrow \mathtt{C} \qquad \Gamma \vdash \overline{\mathtt{e}} : \overline{\mathtt{C}} \qquad \overline{\mathtt{C}} \ \mathtt{<:}\ \overline{\mathtt{D}}}{\Gamma \vdash \mathtt{e}_0.\mathtt{m}(\overline{\mathtt{e}}) : \mathtt{C}}$$

$$\frac{\mathit{fields}(\mathtt{C}) = \overline{\mathtt{D}}\ \overline{\mathtt{f}} \qquad \Gamma \vdash \overline{\mathtt{e}} : \overline{\mathtt{C}} \qquad \overline{\mathtt{C}} \ \mathtt{<:}\ \overline{\mathtt{D}}}{\Gamma \vdash \mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}}) : \mathtt{C}}$$

$$\frac{\Gamma \vdash \mathtt{e}_0 : \mathtt{D}}{\Gamma \vdash (\mathtt{C})\mathtt{e}_0 : \mathtt{C}}$$

Reductions

$$\frac{\mathit{fields}(\mathtt{C}) = \overline{\mathtt{C}}\ \overline{\mathtt{f}}}{(\mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}})).\mathtt{f}_i \longrightarrow \mathtt{e}_i}$$

$$\frac{\mathit{mbody}(\mathtt{m}, \mathtt{C}) = (\overline{\mathtt{x}}, \mathtt{e}_0)}{(\mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}})).\mathtt{m}(\overline{\mathtt{d}}) \longrightarrow [\overline{\mathtt{d}}/\overline{\mathtt{x}}, \mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}})/\mathtt{this}]\mathtt{e}_0}$$

$$\frac{\mathtt{C} <: \mathtt{D}}{(\mathtt{D})(\mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}})) \longrightarrow \mathtt{new}\ \mathtt{C}(\overline{\mathtt{e}})}$$

Part V

Conclusions

My career in a nutshell

- Haskell
- Java
- XML
- Logic

My career in a nutshell

- Theory

- Practice

My career in a nutshell

- Theory
  Generic Java

- Practice
  Featherweight Java

My career in a nutshell

- Theory into Practice
  Generic Java

- Practice into Theory
  Featherweight Java