Justin Bailey jgbailey@codeslower.com

Portland State University

August 6, 2012

Justin Bailey

Introduction

MIL

**Dataflow Analysi** 

Uncurryin

Conclusion

Appendia

## Introduction

- Compiling Functional Programming Languages
- Dataflow Analysis
- ▶ Three-Address Code

Justin Bailey

Introduction

MIL

Dataflow Analysis

Incurrying

Conclusion

Appendi:

# Dataflow Analysis for Functional Languages?

- Can we apply functional-language specific optimizations?
- Can we implement traditional dataflow-based optimizations?

Justin Bailey

Introduction

MIL

Dataflow Analysis

Uncurrying

Conclusio

Appendi

# Monadic Programming to the Rescue!

- ► Simple control-flow
- Separate side-effecting computation from "pure" values
- Higher-order functions

Justin Bailey

Introduction

MIL

Dataflow Analysis

Uncurrying

Conclusion

Appendi:

# MIL: A Monadic Intermediate Language

▶ Monadic: Haskell's **do** notation.

► Monadic: Segregated side-effects.

▶ Dataflow: Basic blocks.

► Dataflow: Block scope.

▶ Dataflow: Based on three-address code.

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

Uncurrying

Conclusion

Appendi:

#### Contributions

- ▶ Applied the *dataflow algorithm* to a functional language.
- ▶ Implemented *uncurrying*, using the dataflow algorithm.
- ► Thorough exposition of the HOOPL library.

Justin Bailey

Introduction

MIL Goals of MIL

Dataflow Analys

Uncurryi

Conclusio

Appendix

## Goals of MIL

- Simplicity
- ► Allocation & Other Side-Effects
- ► Higher-Order Functions

Justin Bailey

Introduction

MIL Goals of MIL

**Dataflow Analysi** 

Officulty

Conclusio

Appendix

# Simplicity

$$\frac{(b*c+d)}{2}$$

Justin Bailey

Introduction

MIL Goals of MIL

Related Work

Datatiow Analys

Onloanyn

Conclusio

Appendi

# Simplicity

$$\frac{(b*c+d)}{2}$$

► Three-Address Code

```
t1 := b * c;
```

Justin Bailey

Introduction

Goals of MIL

neialeu Work

Dalallow Allalys

. . . .

Conclusio

Appendi:

# Simplicity

$$\frac{(b*c+d)}{2}$$

- ► MIL
  - 1 t1 <- mul\*(b, c)
  - 2 t2 <- add\*(t1, d)</pre>
  - $_3$  t3 <- div $^*$ (t2, 2)

Justin Bailey

Introduction

MIL Goals of MIL

Dataflow Analysi

Officultyill

Conclusio

Appendix

## Side-Effects

Justin Bailey

Introduction

MIL Goals of MIL

.....

Datanow / maryo

-----,...

Conclusio

Appendix

## Side-Effects

► Closures

$$\mathtt{t1} \leftarrow \mathtt{k} \left\{ \mathtt{x} \right\}$$

Justin Bailey

Introduction

MIL Goals of MIL

\_ . . . . .

.....

0----

Conclusio

Appendix

## Side-Effects

► Closures

$$t1 \leftarrow k \{x\}$$

Justin Bailey

Introduction

MIL Goals of MIL

\_ . . . . . .

.....

0----

00110100101

#### Side-Effects

Closures

$$t1 \leftarrow k \{x\}$$

▶ Data values

Justin Bailey

Introduction

MIL Goals of MIL

\_ . . . . . .

.....

0----

00110100101

#### Side-Effects

Closures

$$t1 \leftarrow k \{x\}$$

▶ Data values

Justin Bailey

Introduction

MIL Goals of MIL

Dataflau Analusi

,

\_ . . .

Conclusion

Appendia

## Side-Effects

Closures

$$t1 \leftarrow k \{x\}$$

▶ Data values

Primitives

$$\texttt{t3} \leftarrow \texttt{add}^*(\texttt{x},\texttt{y})$$

Justin Bailey

Introduction

MIL Goals of MIL

Detefless Analys

Uncurryi

Canaluaia

Appendi:

# Sufficiency

- ► Higher-Order Functions
- Primitives
- ▶ Data Values

Introduction

MIL

Goals of MIL Related Work

Dataflow Analysis

Officultying

Conclusion

## Related Work

- ▶ MLj: Benton, Kennedy, & Russell¹
- ► Continuation-Passing Style<sup>2</sup>
- ► Administrative-Normal Form: Flanagan, Sabry, Duba, and Felleisen<sup>3</sup>

<sup>&</sup>lt;sup>1</sup>"Compiling Standard ML to Java Bytecodes" (1998).

<sup>&</sup>lt;sup>2</sup>See Appel's "Compiling with Continuations" (1992).

<sup>&</sup>lt;sup>3</sup>"The Essence of Compiling with Continuations" (1993).

Justin Bailey

Introduction

MIL

Dataflow Analysis

Fundamenta Hoopl

Uncurryin

Conclusio

Appendi

# Dataflow Analysis

- Due to Kildall's "A Unified Approach to Global Program Optimization" (1973)
- Widely applied to imperative programming languages

Justin Bailey

Introduction

MIL

Dataflow Analysis
Fundamentals

Uncurryin

Conclusio

Appendi

# Typical Dataflow Optimizations

- ▶ Dead-code Elimination
- Constant Folding
- Lazy Code Motion
- ► For more, see Muchnick's "Advanced compiler design and implementation" (1997)

Justin Bailey

Introduction

МП

Dataflow Analysis

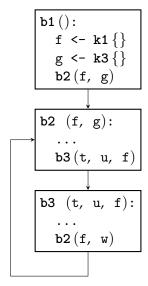
Fundamentals

Uncurrying

Conclusio

Appendi

#### Fundamentals: CFGs & Basic Blocks



**Facts** 

Justin Bailey

Introduction

MIL

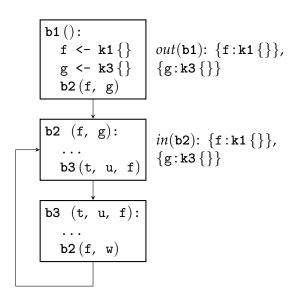
Dataflow Analysi

Fundamentals Hoopl

Oncurrying

Conclusio

Appendi:



Justin Bailey

Introduction

MIL

Dataflow Analysis

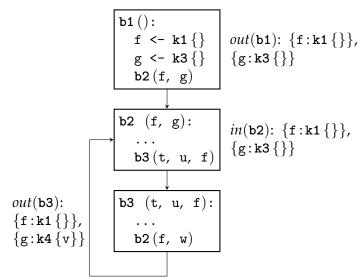
Fundamentals Hoopl

Uncurrying

Conclusio

Appendi:

## **Iteration**



Justin Bailey

Introduction

MIL

Dataflow Analysis

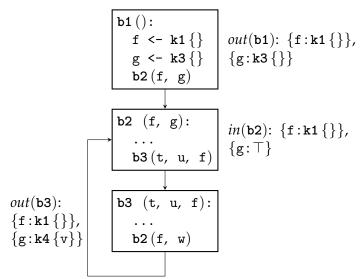
Fundamentals Hoopl

Uncurrying

Conclusio

Appendi

#### **Iteration**



Justin Bailey

Introduction

MIL

Dataflow Analysis
Fundamentals
Hoopl

Uncurryi

Conclusio

Appendi

# HOOPL: A Haskell Library for Dataflow Analysis

- See "Hoopl: A Modular, Reusable Library for Dataflow Analysis and Transformation" by Ramsey, Dias, and Peyton Jones (2010)
- Used in the Glasgow Haskell Compiler
- Based on "Composing Dataflow Analyses and Transformations" by Lerner, Grove, and Chambers (2002)

Justin Bailey

Introduction

MIL

Dataflow Analysis

#### Uncurrying

Uncurrying ma

Conclusio

Appendi:

# Uncurrying

- Partial Application
- Uncurrying map

Justin Bailey

Introduction

MIL

Dataflow Analysis

#### Uncurrying

Related World

Conclusio

Appendi:

# Partial Application

```
map f xs = case xs of

Cons x xs' \rightarrow Cons (f x) (map f xs')

Nil \rightarrow Nil

toList x = Cons x Nil
```

Justin Bailey

Introduction

MIL

Dataflow Analysis

#### Uncurrying

Uncurrying n

Conclusio

Appendi:

# Partial Application

```
mkLists = map \ toList

map \ f \ xs = \mathbf{case} \ xs \ \mathbf{of}

Cons \ x \ xs' \to Cons \ (f \ x) \ (map \ f \ xs')

Nil \to Nil

toList \ x = Cons \ x \ Nil
```

Justin Bailey

Introduction

MIL

Dataflow Analysis

## Uncurrying

Related Work

Conclusio

Appendi

# Partial Application

```
mkLists = map \ toList

main1 \ ns = map \ toList \ ns

main2 \ ns = mkLists \ ns

map \ f \ xs = \mathbf{case} \ xs \ \mathbf{of}

Cons \ x \ xs' \to Cons \ (f \ x) \ (map \ f \ xs')

Nil \to Nil

toList \ x = Cons \ x \ Nil
```

Justin Bailey

Introduction

MIL

Dataflow Analysis

Uncurrying map

Conclusion

Appendi:

```
main ns = map \ to List \ ns

map f \ xs = \mathbf{case} \ xs \ \mathbf{of}

Cons x \ xs' \rightarrow

Cons (f \ x) \ (map \ f \ xs')

Nil \rightarrow Nil

to List x = Cons \ x \ Nil
```

Justin Bailey

Introduction

IVIIL

Dataflow Analysis

Uncurrying map

Conclusion

Appendi

```
\begin{array}{llll} \underline{\textit{main ns}} = \textit{map toList ns} & 1 & \texttt{main (ns):} \\ \textit{map } f \; xs = \mathbf{case} \; xs \; \mathbf{of} & 2 & \texttt{v227} \; <- \; \mathbf{k203} \; \{\} \\ \textit{Cons } x \; xs' \to & 3 & \texttt{v228} \; <- \; \mathbf{k219} \; \{\} \\ \textit{Cons } (f \; x) \; (\textit{map } f \; xs') & 4 & \texttt{v229} \; <- \; \texttt{v227} \; @ \; \texttt{v228} \\ \textit{Nil} \to \textit{Nil} & 5 & \texttt{v229} \; @ \; \text{ns} \\ \textit{toList } x = \textit{Cons } x \; \textit{Nil} \end{array}
```

Justin Bailey

Introduction

MIL

Dataflow Analysis

# Uncurrying map

Related Work

Conclusio

Appendi

```
\mathtt{main}\,(\mathtt{ns})\colon
```

```
main ns = map toList ns1main (ns):map f xs = case xs of2v227 <- k203 {}</th>Cons x xs' \rightarrow3v228 <- k219 {}</th>Cons (f x) (map f xs')4v229 <- v227 @ v228</th>Nil \rightarrow Nil5v229 @ nstoList x = Cons x Nil
```

Justin Bailey

Introduction

MIL

Dataflow Analysis

Uncurrying map

Related Work

Conclusio

Appendi

```
main ns = map \ toList \ ns 1 toList (x):

map f \ xs = \mathbf{case} \ xs \ \mathbf{of} 2 v221 <- Cons1 {}

Cons x \ xs' \rightarrow 3 v222 <- v221 @ x

Cons (f \ x) \ (map \ f \ xs') 4 v223 <- Nil

Nil \rightarrow Nil 5 v222 @ v223

toList x = Cons \ x \ Nil
```

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

Uncurrying map

Related Work

Conclusio

Appendi

```
\mathtt{toList}\,(\mathtt{x})\colon
```

```
main ns = map \ to List \ ns 1 to List (x):

map f \ xs = \mathbf{case} \ xs \ \mathbf{of} 2 v221 <- Cons1 {}

Cons x \ xs' \rightarrow 3 v222 <- v221 @ x

Cons (f \ x) \ (map \ f \ xs') 4 v223 <- Nil

Nil \rightarrow Nil 5 v222 @ v223

to List x = Cons \ x \ Nil
```

Justin Bailey

Introduction

MIL

Dataflow Analysis

Uncurrying map

Conclusion

Appendi:

```
main ns = map \ to List \ ns1map \ (f, xs):map \ f \ xs = case \ xs \ of2case \ xs \ ofCons \ x \ xs' \rightarrow3Nil \ -> nil \ ()Cons \ (f \ x) \ (map \ f \ xs')4Cons \ x \ xs \ ->Nil \ \rightarrow Nil5cons \ (f, \ x, \ xs)to List \ x = Cons \ x \ Nil
```

Justin Bailey

Introduction

MIL

Dataflow Analysis

Uncurrying map

0----

Appendi

Justin Bailey

Introduction

MIL

Dataflow Analysis

Uncurrying map

Conclusio

Append

```
map(f,xs):
nil():
                                cons(f,x,xs):
                                cons(f, x, xs):
main ns = map to List ns
map f xs = case xs of
                                  v209 <- Cons1 { }
  Cons x xs' \rightarrow
                                  v210 < - f @ x
    Cons (f x) (map f xs')
                                  v211 <- v209 @ v210
  Nil \rightarrow Nil
                                  v212 <- k203 {}
toList x = Cons x Nil
                                  v213 < - v212 @ f
                                  v214 < - v213 @ xs
                             7
                                  v211 @ v214
```

Justin Bailey

Introduction

MIL

Dataflow Analysis

Uncurrying map

Conclusio

Appendi:

```
mil(): map(f,xs): cons(f,x,xs): main ns = map to List ns nil(): Nil map f xs = case xs of cons(fx) (map f xs') Nil \rightarrow Nil to List x = Cons x Nil
```

Justin Bailey

Introduction

IVIIL

Dataflow Analysis

Uncurrying map
Related Work

Conclusio

Appendi:

# Uncurrying map

toList x = Cons x Nil



$$mil(): \leftarrow map(f,xs): \rightarrow cons(f,x,xs):$$
 $main \ ns = map \ toList \ ns$ 
 $map \ f \ xs = case \ xs \ of$ 
 $Cons \ x \ xs' \rightarrow Cons(f \ x) \ (map \ f \ xs')$ 
 $Nil \rightarrow Nil$ 

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

### Uncurrying map

Uncurrying ma Related Work

Conclusio

Appendi

```
main(ns): toList(x):
```

```
\min(): \longleftarrow \max(f,xs): \longrightarrow \cos(f,x,xs):
```

```
1 main (ns):
```

$$7 \text{ k204} \{f\} xs: map(f,xs)$$

$$s$$
 k219  $\{\}$  x: toList  $(x)$ 

Justin Bailey

Introduction

MIL

Dataflow Analysis

### Uncurrying map

Related Work

Conclusio

Appendi

```
\boxed{ \texttt{main} \, (\texttt{ns}) \colon } \qquad \boxed{ \texttt{toList} \, (\texttt{x}) \colon }
```

Justin Bailey

Introduction

MIL

Dataflow Analysis

### Uncurrying map

Related Work

Conclusio

Appendi

```
\boxed{ \texttt{main} \, (\texttt{ns}) \colon } \qquad \boxed{ \texttt{toList} \, (\texttt{x}) \colon }
```

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

### Uncurrying map

Related Work

Conclusio

Appendi

```
\boxed{ \texttt{main} \, (\texttt{ns}) \colon } \qquad \boxed{ \texttt{toList} \, (\texttt{x}) \colon }
```

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

Uncurrying map

Canalusia

Appendi

```
\boxed{ \texttt{main} \, (\texttt{ns}) \colon } \qquad \boxed{ \texttt{toList} \, (\texttt{x}) \colon }
```

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

### Uncurrying map

Related Work

Conclusio

Appendi

```
\boxed{ \texttt{main} \, (\texttt{ns}) \colon } \qquad \boxed{ \texttt{toList} \, (\texttt{x}) \colon }
```

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

### Uncurrying map

Uncurrying ma

Conclusio

Appendi

```
\boxed{ \texttt{main} \, (\texttt{ns}) \colon } \qquad \boxed{ \texttt{toList} \, (\texttt{x}) \colon }
```

```
\boxed{ \mathtt{nil}() \colon} \longleftarrow \boxed{ \mathtt{map}(\mathtt{f},\mathtt{xs}) \colon} \longrightarrow \boxed{ \mathtt{cons}(\mathtt{f},\mathtt{x},\mathtt{xs}) \colon}
```

```
_{1} \quad \mathtt{main}\,(\mathtt{ns}) \colon \longleftarrow \{\mathtt{ns} \colon \top\}
```

$$v227 \leftarrow k203\{\} \leftarrow \{v227: k203\{\}\}$$

$$v228 \leftarrow k219\{\} \leftarrow \{v228:k219\{\}\}$$

$$\begin{tabular}{lll} & & & v229 & \mbox{$<$-$} & & k204 & \{v228\} & \mbox{$\leftarrow$-$} & \{v229: k204 & \{v228\}\} \\ \end{tabular}$$

- v229 @ ns
- 6 k203 {} f: k204 {f}
- 7  $k204 \{f\} xs: map(f,xs)$
- s k219  $\{\}$  x: toList (x)

Justin Bailey

Introduction

**Dataflow Analysis** 

## Uncurrying may

# Uncurrying map

k219 {} x: toList(x)

```
main(ns):
                toList(x):
```

```
map(f,xs):
nil():
                                       cons(f,x,xs):
   main(ns): \leftarrow \{ns: \top\}
      v227 \leftarrow k203\{\} \leftarrow \{v227: k203\{\}\}
      v228 \leftarrow k219 \{ \} \leftarrow \{ v228 : k219 \{ \} \}
3
      v229 \leftarrow k204 \{v228\} \leftarrow \{v229: k204 \{v228\}\}
      v229 @ ns
   k203{}f: k204{f}
   k204 \{f\} xs: map(f,xs)
```

Justin Bailey

Introduction

IVIIL

**Dataflow Analysis** 

# Uncurrying Uncurrying map

Related Work

Conclusio

Appendi

```
\boxed{ \texttt{main} \, (\texttt{ns}) \colon } \qquad \boxed{ \texttt{toList} \, (\texttt{x}) \colon }
```

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

### Uncurrying Uncurrying map

Related Work

Conclusio

Appendi

```
\boxed{ \texttt{main} \, (\texttt{ns}) \colon } \qquad \boxed{ \texttt{toList} \, (\texttt{x}) \colon }
```

```
\begin{array}{lll} & \min{(\text{ns}):} & \longleftarrow \{\text{ns}:\top\} \\ & \text{v227} & \leftarrow \{\text{k203}\}\} & \leftarrow \{\text{v227}:\text{k203}\}\} \\ & \text{v228} & \leftarrow \text{k219}\}\} & \leftarrow \{\text{v228}:\text{k219}\}\} \\ & \text{v229} & \leftarrow \text{k204}\{\text{v228}\} & \leftarrow \{\text{v229}:\text{k204}\{\text{v228}\}\}\} \\ & \text{k204}\{\text{v228}\} & \text{0 ns} \\ & \text{6 k203}\}\} & \text{f: k204}\{f\} \\ & \text{7 k204}\{f\} & \text{xs: map}(f,\text{xs}) \\ & \text{8 k219}\} & \text{x: toList}(x) \end{array}
```

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

# Uncurrying map

Related Work

Conclusio

Appendi

```
\boxed{ \texttt{main} \, (\texttt{ns}) \colon } \qquad \boxed{ \texttt{toList} \, (\texttt{x}) \colon }
```

```
\begin{array}{lll} & \min{(\text{ns}):} & \longleftarrow \{\text{ns}:\top\} \\ & \text{v227} & \leftarrow \{\text{k203}\{\} & \longleftarrow \{\text{v227}:\text{k203}\{\}\} \\ & \text{v228} & \leftarrow \text{k219}\{\} & \longleftarrow \{\text{v228}:\text{k219}\{\}\} \\ & \text{v229} & \leftarrow \text{k204}\{\text{v228}\} & \longleftarrow \{\text{v229}:\text{k204}\{\text{v228}\}\} \\ & \text{map}(\text{v228}, \text{ns}) \\ & \text{k203}\{\} \text{ f: k204}\{\text{f}\} \\ & \text{v229} & \text{k204}\{\text{f}\} \text{ xs: map}(\text{f,xs}) \\ & \text{k219}\{\} \text{ x: toList}(\text{x}) \end{array}
```

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

### Uncurrying map

Related Work

Conclusio

Appendi

```
Uncurrying map
                  main(ns):
                                        toList(x):
                 map(f,xs):
  nil():
                                       cons(f,x,xs):
     main(ns): \leftarrow \{ns: \top\}
        v227 \leftarrow k203\{\} \leftarrow \{v227: k203\{\}\}
        v228 \leftarrow k219 \{ \} \leftarrow \{ v228 : k219 \{ \} \}
 3
        v229 \leftarrow k204 \{v228\} \leftarrow \{v229: k204 \{v228\}\}
        map (v228, ns)
     k203\{\}f: k204\{f\}
     k204 \{f\} xs: map(f,xs)
     k219 {} x: toList(x)
```

Justin Bailey

Introduction

MIL

Dataflow Analysis

## Uncurrying map

Related Work

Conclusio

Appendi

```
Uncurrying map
                 main(ns):
                                       toList(x):
                 map(f,xs):
  nil():
                                     cons(f,x,xs):
     main(ns): \leftarrow \{ns: \top\}
       #14/211//KH/14/20/3/{}}
       v228 \leftarrow k219 \{ \} \leftarrow \{ v228 : k219 \{ \} \}
 3
       #121211/KH/142104/{#12128}}
       map (v228, ns)
     k203\{\}f: k204\{f\}
     k204 \{f\} xs: map(f,xs)
```

k219 {} x: toList(x)

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

### Uncurrying map

Related Work

Conclusio

Appendi:

```
Uncurrying map
                 main(ns):
                                      toList(x):
                map(f,xs):
                                    cons(f,x,xs):
  nil():
     main(ns): \leftarrow \{ns: \top\}
       v228 \leftarrow k219 \{ \} \leftarrow \{ v228 : k219 \{ \} \}
       map (v228, ns)
 3
     k203\{\}f: k204\{f\}
     k204 \{f\} xs: map(f,xs)
```

k219 {} x: toList(x)

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

### Uncurrying map

Related Work

Conclusio

Appendi:

```
 \begin{array}{c|c} & & \\ \hline \text{main} \, (\text{ns}) \colon & \\ \hline & & \\ & & \\ \hline & in(\text{map}) \colon \{\text{f:k219}\, \{\}\}, \, \{\text{xs:}\, \top\} \\ \hline \\ \hline & \text{nil} \, () \colon & \\ \hline \end{array}
```

- $_1$  main (ns):
- 2 v228 <- **k219** {}
  - $_3$  map (v228, ns)
  - 4 k203 {} f: k204 {f}
  - 5  $k204 \{f\} xs: map (f, xs)$
- 6 k219 {} x: toList(x)

Justin Bailey

Introduction

**Dataflow Analysis** 

Uncurrying map

```
main(ns):
                                 toList(x):
                       in(map): \{f: k219 \{\}\}, \{xs: \top\}
             map(f,xs):
                                cons(f,x,xs):
nil():
```

- $\begin{array}{c} \texttt{map}\,(\texttt{f},\texttt{xs}) \colon & \\ \texttt{case} \ \texttt{xs} \ \texttt{of} \end{array} \{ \texttt{f} \colon \texttt{k219}\, \{ \} \}, \, \{\texttt{xs} \colon \top \} \end{array}$ 2
- Nil -> nil() 3
- Cons  $x xs \rightarrow cons(f, x, xs)$  $\{x:\top\}, \{xs:\top\}$

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

### Uncurrying

Uncurrying ma

Conclusio

Appendi

```
main(ns):
                                     toList(x):
                          in(map): \{f: k219 \{\}\}, \{xs: \top\}
      in(nil):\emptyset
               map(f,xs):
                                    cons(f,x,xs):
nil():
                in(cons): \{f: k219 \{\}\}, \{x: \top\}, \{xs: \top\}
   map(f,xs):
                  \{f: k219 \{\}\}, \{xs: \top\}
2
        Nil -> nil()
3
        Cons x xs \rightarrow cons(f, x, xs)
                  \{x:\top\},\{xs:\top\}
```

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

### Uncurrying

Uncurrying map

Conclusion

Appendi

```
in(nil): \emptyset \qquad in(map): \{f: k219 \{\}\}, \{xs: \top\}
nil(): \longleftarrow map(f, xs): \longrightarrow cons(f, x, xs):
in(cons): \{f: k219 \{\}\}, \{x: \top\}, \{xs: \top\}
```

- 1 cons (f, x, xs):
- 2 v209 <- Cons1 {}
- 3 v210 <- f @ x
- 4 v211 <- v209 @ v210
- 5 v212 <- k203 {}
- 6 v213 <- v212 @ f
- 7 v214 <- v213 @ xs
- 8 v211 @ v214

Justin Bailey

Introduction

**Dataflow Analysis** 

Uncurrying

Related Work

Conclusion

Appendix

# Uncurrying map

v211 @ v214

```
1 cons (f, x, xs): ← {f:k219 {}}, {x: ⊤}, {xs: ⊤}

2 v209 <- Cons1 {}

3 v210 <- f @ x

4 v211 <- v209 @ v210

5 v212 <- k203 {}

6 v213 <- v212 @ f

7 v214 <- v213 @ xs
```

Justin Bailey

Introduction

**Dataflow Analysis** 

Uncurrying
Uncurrying may

Uncurrying ma

Conclusion

Appendix

# Uncurrying map

v211 @ v214

```
1 cons (f, x, xs): ← {f:k219 {}}, {x: ⊤}, {xs: ⊤}

2 v209 <- Cons1 {} ← {v209:Cons1 {}}

3 v210 <- f @ x

4 v211 <- v209 @ v210

5 v212 <- k203 {}

6 v213 <- v212 @ f

7 v214 <- v213 @ xs
```

Justin Bailey

Introduction

**Dataflow Analysis** 

Uncurrying

Uncurrying ma

Conclusio

Appendix

```
1 cons (f, x, xs): ←{f:k219{}}, {x:⊤}, {xs:⊤}

2 v209 <- Cons1{} ←{v209:Cons1{}}

3 v210 <- f @ x

4 v211 <- v209 @ v210

5 v212 <- k203{}

6 v213 <- v212 @ f

7 v214 <- v213 @ xs

8 v211 @ v214

9 k219{}x: toList(x)
```

```
Using Dataflow
Optimization
Techniques with a
Monadic
Intermediate
Language
```

Justin Bailey

Introduction

**Dataflow Analysis** 

Uncurrying map

Related Work

Conclusio

Appendia

```
1 cons (f, x, xs): ← {f:k219 {}}, {x: ⊤}, {xs: ⊤}

2 v209 <- Cons1 {} ← {v209:Cons1 {}}

3 v210 <- k219 {} @ x

4 v211 <- v209 @ v210

5 v212 <- k203 {}

6 v213 <- v212 @ f

7 v214 <- v213 @ xs

8 v211 @ v214

9 k219 {} x: toList (x)
```

Justin Bailey

Introduction

Dataflow Analysis

Uncurrying

Uncurrying map

Conclusion

Appendix

```
cons (f, x, xs): ←{f:k219{}}, {x:⊤}, {xs:⊤}
v209 <- Cons1{} ←{v209:Cons1{}}
v210 <- toList(x)
v211 <- v209 @ v210
v212 <- k203{}
v213 <- v212 @ f
v214 <- v213 @ xs
v211 @ v214
k219{}x: toList(x)</pre>
```

```
Using Dataflow
  Optimization
Techniques with a
    Monadic
  Intermediate
   Language
```

Justin Bailey

Introduction

**Dataflow Analysis** 

10

### Uncurrying may

```
cons (f, x, xs): \leftarrow \{f: k219 \{\}\}, \{x: \top\}, \{xs: \top\}\}
      v209 \leftarrow Cons1\{\} \leftarrow \{v209 : Cons1\{\}\}\}
      v210 \leftarrow toList(x) \leftarrow \{v210: \top\}
      v211 <- v209 @ v210
     v212 <- k203 {}
      v213 <- v212 @ f
      v214 <- v213 @ xs
7
      v211 @ v214
8
   Cons1\{\}a2: Cons2\{a2\}
   Cons2 {a2} a1: Cons a2 a1
```

```
Using Dataflow
Optimization
Techniques with a
Monadic
Intermediate
Language
```

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

### Uncurrying map

Uncurrying ma

Conclusion

. ..

```
1 cons(f, x, xs): \leftarrow \{f:k219\{\}\}, \{x:\top\}, \{xs:\top\}\}

2 v209 \leftarrow Cons1\{\} \leftarrow \{v209:Cons1\{\}\}\}

3 v210 \leftarrow toList(x) \leftarrow \{v210:\top\}

4 v211 \leftarrow Cons1\{\} @ v210

5 v212 \leftarrow k203\{\}

6 v213 \leftarrow v212 @ f

7 v214 \leftarrow v213 @ xs

8 v211 @ v214

9 Cons1\{\} a2: Cons2\{a2\}

10 Cons2\{a2\} a1: Cons a2 a1
```

Justin Bailey

Introduction

MIL

Dataflow Analysis

### Uncurrying map

Uncurrying ma

Conclusio

```
1 cons(f, x, xs): \leftarrow \{f:k219\{\}\}, \{x:\top\}, \{xs:\top\}\}

2 v209 \leftarrow Cons1\{\} \leftarrow \{v209:Cons1\{\}\}\}

3 v210 \leftarrow toList(x) \leftarrow \{v210:\top\}

4 v211 \leftarrow Cons2\{v210\}

5 v212 \leftarrow k203\{\}

6 v213 \leftarrow v212 @ f

7 v214 \leftarrow v213 @ xs

8 v211 @ v214

9 Cons1\{\}a2: Cons2\{a2\}

10 Cons2\{a2\}a1: Cons a2 a1
```

Justin Bailey

Introduction

IVIIL

Dataflow Analysis

### Uncurrying map

Uncurrying ma

Conclusio

00..0.00.0

Appendia

```
1 cons(f, x, xs): \leftarrow \{f:k219\{\}\}, \{x:\top\}, \{xs:\top\}\}

2 v209 \leftarrow Cons1\{\} \leftarrow \{v209:Cons1\{\}\}\}

3 v210 \leftarrow toList(x) \leftarrow \{v210:\top\}

4 v211 \leftarrow Cons2\{v210\} \leftarrow \{v211:Cons2\{v210\}\}\}

5 v212 \leftarrow k203\{\} \leftarrow \{v212:k203\{\}\}\}

6 v213 \leftarrow v212 \ f \leftarrow \{v213:\top\}

7 v214 \leftarrow v213 \ xs \leftarrow \{v214:\top\}

8 v211 \ v214

9 v214 \leftarrow v214

10 v214 \leftarrow v214

11 v214 \leftarrow v214

12 v214 \leftarrow v214
```

Justin Bailey

Introduction

....

Uncurrying

Uncurrying map

Conclusio

Appendia

```
1 cons (f, x, xs): \leftarrow {f:k219 {}}, {x: \top}, {xs: \top}
2 v209 <- Cons1 {} \leftarrow {v209:Cons1 {}}
3 v210 <- toList (x) \leftarrow {v210: \top}
4 v211 <- Cons2 {v210} \leftarrow {v211:Cons2 {v210}}
5 v212 <- k203 {} \leftarrow {v212:k203 {}}
6 v213 <- k203 {} @ f \leftarrow {v213: \top}
7 v214 <- v213 @ xs \leftarrow {v214: \top}
8 v211 @ v214
9 k203 {} f: k204 {f}
10 k204 {f} xs: map (f, xs)
```

Justin Bailey

Introduction

MIL

Dataflow Analysis

### Uncurrying map

Uncurrying ma

Canalucia

Conclusio

Appendi:

```
1 cons(f, x, xs): \leftarrow \{f:k219\{\}\}, \{x:\top\}, \{xs:\top\}\}

2 v209 \leftarrow Cons1\{\} \leftarrow \{v209:Cons1\{\}\}\}

3 v210 \leftarrow toList(x) \leftarrow \{v210:\top\}

4 v211 \leftarrow Cons2\{v210\} \leftarrow \{v211:Cons2\{v210\}\}\}

5 v212 \leftarrow k203\{\} \leftarrow \{v212:k203\{\}\}\}

6 v213 \leftarrow k204\{f\} \leftarrow \{v213:k204\{f\}\}\}

7 v214 \leftarrow v213 @ xs \leftarrow \{v214:\top\}

8 v211 @ v214

9 v214 \leftarrow v214

10 v214 \leftarrow v214

11 v214 \leftarrow v214

12 v214 \leftarrow v214
```

Justin Bailey

Introduction

....\_

Dataflow Analysis

### Uncurrying

Uncurrying ma

Conclusio

Coriciasio

Appendi:

```
1 cons (f, x, xs): \leftarrow {f:k219 {}}, {x:\top}, {xs:\top}
2 v209 <- Cons1 {} \leftarrow {v209:Cons1 {}}
3 v210 <- toList (x) \leftarrow {v210:\top}
4 v211 <- Cons2 {v210} \leftarrow {v211:Cons2 {v210}}
5 v212 <- k203 {} \leftarrow {v212:k203 {}}
6 v213 <- k204 {f} \leftarrow {v213:k204 {f}}
7 v214 <- v213 @ xs \leftarrow {v214:\top}
8 v211 @ v214
9 k203 {} f: k204 {f}
10 k204 {f} xs: map (f, xs)
```

Justin Bailey

Introduction

IVIIL

Dataflow Analysis

### Uncurrying map

Uncurrying ma

Conclusion

Conclusio

Appendi:

```
1 cons(f, x, xs): \leftarrow \{f:k219\{\}\}, \{x:\top\}, \{xs:\top\}\}

2 v209 \leftarrow Cons1\{\} \leftarrow \{v209:Cons1\{\}\}\}

3 v210 \leftarrow toList(x) \leftarrow \{v210:\top\}

4 v211 \leftarrow Cons2\{v210\} \leftarrow \{v211:Cons2\{v210\}\}\}

5 v212 \leftarrow k203\{\} \leftarrow \{v212:k203\{\}\}\}

6 v213 \leftarrow k204\{f\} \leftarrow \{v213:k204\{f\}\}\}

7 v214 \leftarrow k204\{f\} \otimes xs \leftarrow \{v214:\top\}

8 v211 \otimes v214

9 v214 \leftarrow k204\{f\}

10 v214 \leftarrow k204\{f\}

11 v214 \leftarrow k204\{f\}
```

Justin Bailey

Introduction

### Uncurrying may

```
cons (f, x, xs): \leftarrow \{f: k219 \{\}\}, \{x: \top\}, \{xs: \top\}\}
       v209 \leftarrow Cons1\{\} \leftarrow \{v209:Cons1\{\}\}\}
       v210 \leftarrow toList(x) \leftarrow \{v210: \top\}
3
       v211 \leftarrow Cons2\{v210\} \leftarrow \{v211:Cons2\{v210\}\}\
       v212 \leftarrow k203\{\} \leftarrow \{v212:k203\{\}\}
       v213 \leftarrow k204\{f\} \leftarrow \{v213: k204\{f\}\}\
       v214 \leftarrow map(f,xs) \leftarrow \{v214:\top\}
7
       v211 @ v214
8
    k203 {} f: k204 {f}
    k204 \{f\} xs: map(f, xs)
10
```

Justin Bailey

Introduction

### Uncurrying may

```
cons (f, x, xs): \leftarrow \{f: k219 \{\}\}, \{x: \top\}, \{xs: \top\}\}
       v209 \leftarrow Cons1\{\} \leftarrow \{v209:Cons1\{\}\}\}
       v210 \leftarrow toList(x) \leftarrow \{v210: \top\}
       v211 \leftarrow Cons2\{v210\} \leftarrow \{v211:Cons2\{v210\}\}
       v212 \leftarrow k203\{\} \leftarrow \{v212:k203\{\}\}
       v213 \leftarrow k204\{f\} \leftarrow \{v213: k204\{f\}\}\
       v214 \leftarrow map(f,xs) \leftarrow \{v214:\top\}
7
       v211 @ v214
8
    Cons1 { } a2: Cons2 { a2 }
    Cons2 {a2} a1: Cons a2 a1
10
```

Justin Bailey

Introduction

## Uncurrying may

```
cons (f, x, xs): \leftarrow \{f: k219 \{\}\}, \{x: \top\}, \{xs: \top\}\}
       v209 \leftarrow Cons1\{\} \leftarrow \{v209:Cons1\{\}\}\}
       v210 \leftarrow toList(x) \leftarrow \{v210: \top\}
       v211 \leftarrow Cons2\{v210\} \leftarrow \{v211: Cons2\{v210\}\}
       v212 \leftarrow k203\{\} \leftarrow \{v212:k203\{\}\}
       v213 \leftarrow k204\{f\} \leftarrow \{v213: k204\{f\}\}\
       v214 \leftarrow map(f,xs) \leftarrow \{v214:\top\}
7
       Cons2 {v210} @ v214
8
    Cons1\{\}a2: Cons2\{a2\}
    Cons2 {a2} a1: Cons a2 a1
10
```

Justin Bailey

Introduction

## Uncurrying may

```
cons (f, x, xs): \leftarrow \{f: k219 \{\}\}, \{x: \top\}, \{xs: \top\}\}
       v209 \leftarrow Cons1\{\} \leftarrow \{v209:Cons1\{\}\}\}
       v210 \leftarrow toList(x) \leftarrow \{v210: \top\}
       v211 \leftarrow Cons2\{v210\} \leftarrow \{v211:Cons2\{v210\}\}\
       v212 \leftarrow k203\{\} \leftarrow \{v212:k203\{\}\}
       v213 \leftarrow k204\{f\} \leftarrow \{v213: k204\{f\}\}\
       v214 \leftarrow map(f,xs) \leftarrow \{v214:\top\}
7
       Cons v210 v214
8
    Cons1\{\}a2: Cons2\{a2\}
    Cons2 {a2} a1: Cons a2 a1
10
```

Justin Bailey

Introduction

IVIIL

Dataflow Analysis

### Uncurrying map

Uncurrying ma

Conclusion

Appendix

```
cons (f, x, xs):
```

- v209 <- Cons1 {}
- $v210 \leftarrow f @ x$
- 4 v211 <- v209 @ v210
- 5 v212 <- **k203** {}
- $_{6}$  v213 <- v212 @ f
- 7 v214 <- v213 @ xs
- s v211 @ v214

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

### Uncurrying

Uncurrying ma

. ....

Conclusio

Appendi

```
cons (f, x, xs):
v209 <- Cons1 {}
v210 <- toList (x)
v211 <- Cons2 {v210}
v212 <- k203 {}
v213 <- k204 {f}
v214 <- map (f, xs)
Cons v210 v214</pre>
```

Justin Bailey

Introduction

MIL

Dataflow Analysis

#### Uncurrying Uncurrying map

Uncurrying m

Conclusio

00..0.00.0

Appendi:

```
ons (f, x, xs):
```

- 2 #12/019/14H/12/614\$1/{/}
- $v210 \leftarrow toList(x)$
- 4 \$1211N/KH/Q\$dd\$2/{K2NQ}}
  - #121121/KH/1**4203**/{}}
- 6 #12124/KH/142104/{£/}
- $v214 \leftarrow map(f,xs)$
- 8 Cons v210 v214

Justin Bailey

Introduction

MIL

Dataflow Analysis

## Uncurrying map

Related Work

Conclusio

Appendi:

```
1 cons (f, x, xs):
```

- v210 <- toList(x)</pre>
- $v214 \leftarrow map(f,xs)$
- 4 Cons v210 v214

Justin Bailey

Introduction

MII

Dataflow Analysis

### Uncurrying map

Related Work

Conclusio

Append

# Uncurrying map

 $\boxed{ \texttt{main} \, (\texttt{ns}) \colon } \qquad \boxed{ \texttt{toList} \, (\texttt{x}) \colon }$ 

```
\boxed{ \texttt{nil} \, () \colon} \longleftarrow \boxed{ \texttt{map} \, (\texttt{f}, \texttt{xs}) \colon} \longrightarrow \boxed{ \texttt{cons} \, (\texttt{f}, \texttt{x}, \texttt{xs}) \colon}
```

```
1 main (ns):
```

$$v228 \leftarrow k219\{$$

5 v229 @ ns

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

## Uncurrying map

Related Work

Conclusio

Append

```
Uncurrying map

[main(ns): toList(x):]
```

cons(f,x,xs):

map(f,xs):

```
1 main (ns):
2 #DZT//kH//HZDS/{/}
3 v228 <- k219 {}
4 #DZD/kH//HZD7/Q//HZDS
```

map(v228, ns)

**nil**():

5

Justin Bailey

Introduction

MIL

Dataflow Analysis

## Uncurrying map

Related Work

Conclusio

Append

```
\boxed{ \begin{array}{c} \texttt{main}\,(\texttt{ns}) \colon} & \texttt{toList}\,(\texttt{x}) \colon \\ \\ \hline \\ \texttt{nil}\,() \colon & \longleftarrow \\ \boxed{ \begin{array}{c} \texttt{map}\,(\texttt{f},\texttt{xs}) \colon} \\ \\ \hline \end{array} } \end{array} } \begin{array}{c} \texttt{cons}\,(\texttt{f},\texttt{x},\texttt{xs}) \colon \\ \\ \hline \end{array}
```

- cons(f, x, xs):
- 2 v209 <- Cons1 {}
- 3 v210 <- f @ x
- 4 v211 <- v209 @ v210
- 5 v212 <- k203 {}
- 6 v213 <- v212 @ f
- v214 <- v213 @ xs</pre>
- s v211 @ v214

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

## Uncurrying map

Uncurrying ma

Conclusio

Append

```
        main (ns):
        toList(x):

        nil():
        cons(f,x,xs):
```

- 1 cons(f, x, xs):
- 2 #12/012/1/1/1/06/1/\$1/{/}
- $v210 \leftarrow toList(x)$
- 4 #12111/KH/H2N9/Q/K12110
- 5 #1**21121/KH/142103/{**}
- 6 #12118//KH/H2N2/N2/N4/15/
- v214 < map(f,xs)
- 8 Cons v210 v214

Justin Bailey

Introduction

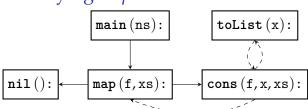
MIL

Dataflow Analysis

Uncurrying map

Conclusio

Appendi



- One closure and two applications eliminated from main.
- Link from main to map.
- ► Two closures and four function applications from cons.
- ▶ Link from cons to map and toList.

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

Uncurrying mu

Conclusio

Appendix

### Related Work

- ▶ Appel: Uncurrying by pattern matching; "Compiling with Continuations" (1992)
- ➤ Tarditi: Uncurrying in four passes; "Design and Implementation of Code Optimizations for a Type-Directed Compiler for Standard ML" (1996)
- ► Tolmach & Oliva: Automatic uncurrying; "From ML to Ada: Strongly-typed Language Interoperability via Source Translation" (1998)

Justin Bailey

Introduction

MIL

Dataflow Analysis

Uncurrying

### Conclusion

Monadic Optimization:

Appendi

## Conclusion

- Monadic Optimizations
- Contributions

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

Uncurrying

Conclusion

Monadic Optimizations

Contributions

Appendi

# Optimizing using the Monad Laws

▶ *Left-Unit* 

$$\mathbf{do} \{x \leftarrow return \ y; m\} \equiv \mathbf{do} \{[y \mapsto x] \ m\}$$

▶ Right-Unit

$$\mathbf{do} \{x \leftarrow m; return \ x\} \equiv \mathbf{do} \{m\}$$

► Associativity

$$\mathbf{do} \{x \leftarrow \mathbf{do} \{y \leftarrow m; n\}; o\} \equiv \mathbf{do} \{y \leftarrow m; x \leftarrow n; o\}$$

► From Wadler's "Monads for Functional Programming" (1995)

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

Uncurrying

Conclusion

Monadic Optimization

Contributions

Append

# Dataflow Analysis & MIL

- ► High-level functional programming; low-level details.
- ► Structured for dataflow analysis.
- ► Implemented other optimizations; for example, dead-code elimination.

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

Uncurrying

Conclusion

Monadic Optimization

Contributions

Appendi

# Uncurrying

- ► Implemented using the dataflow algorithm
- Able to uncurry across blocks and loops (with some caveats)
- ▶ Complete implementation described

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

Uncurrying

Conclusion

Contributions

Appendi

### HOOPL

- Thorough description of the library
- ► Simple, but complete, example implementation given

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

Uncurrying

Conclusion

Monadic Optimization

Contributions

Appendi

### Questions?

Source code and paper available at http://mil.codeslower.com, or email me at jgbailey@codeslower.com.

```
Using Dataflow
Optimization
Techniques with a
Monadic
Intermediate
Language
```

Justin Bailey

```
Introduction
```

MII

Dataflow Analysis

Uncurryin

Conclusi

Appendix
Uncurrying Comparisons
Example: Uncurrying with
Loops

Future Work

# Related Work: Appel

```
f x = g x
g x y = x + y
main \ a \ b = f \ a \ b
  main(a, b): b208(a, b)
  b208 (x, y):
     v210 \leftarrow plusclo1\{x\}
     v210 @ v
  plusclo1{a2}a1: plus*(a2, a1)
  f(): k212{}
  k212\{\}x: k207\{x\}
  g(): k206 {}
  k206\{ \} x: k207\{ x \}
  k207 \{x\} y: b208 (x, y)
```

```
Using Dataflow
  Optimization
Techniques with a
    Monadic
  Intermediate
   Language
```

Justin Bailey

```
Introduction
```

**Dataflow Analysis** 

```
Uncurrying Comparisons
```

## Related Work: Tarditi

```
g x y z = x + y + z
h x y = g x y
main s t u = h s t u
   main(s, t, u): b216(s, t, u)
  b216(x, y, z):
     v219 \leftarrow plusclo1\{x\}
3
```

- v220 <- v219 @ v
- v221 <- plusclo1 {v220}
  - v221 @ z
- $plusclo1{a2}a1: plus*(a2, a1)$
- g{}: k213{}
- $k213\{\}x: k214\{x\}$
- $k214\{x\}y: k215\{x, y\}$ 10
  - h{}: k207{}
  - $k207 \{\} x: k208 \{x\}$
  - $k208 \{x\} y: k215 \{x, y\}$
- $k215 \{x, y\} z: b216 (x, y, z)$

Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

Uncurrying

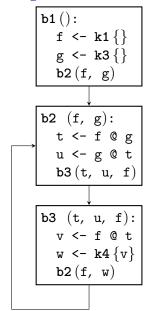
Conclusion

#### Append

Uncurrying Comparisons
Example: Uncurrying with
Loops

Future Work

## Original Program



Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

Uncurrying

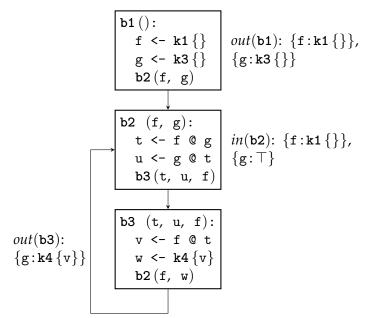
Conclusion

#### Appendi

Uncurrying Comparisons
Example: Uncurrying with
Loops

Future Worl

### **Initial Facts**



Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

Uncurrying

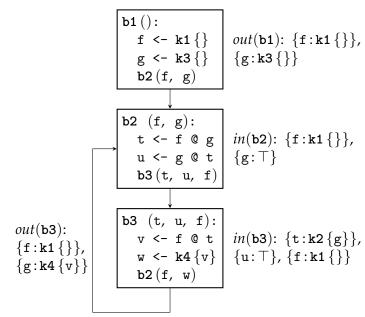
Conclusion

#### Append

Uncurrying Comparisons Example: Uncurrying with Loops

Future Worl

### **Facts After Iteration**



Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

Uncurrying

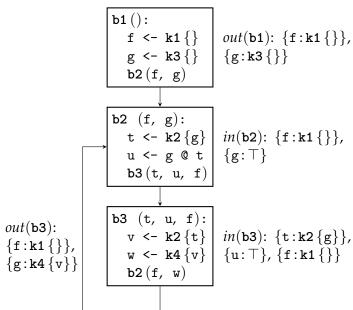
Conclusion

#### Append

Uncurrying Comparisons Example: Uncurrying with Loops

Future Work

## Rewrite



Justin Bailey

Introduction

MIL

**Dataflow Analysis** 

Uncurrying

Conclusio

## Appendix Uncurrying Comparison

Example: Uncurrying with Loops

Future Work

## Future Work

- Eliminating Thunks
- ► "Push Through Cases"