

# Tridirectional Typechecking

Joshua Dunfield and Frank Pfenning

TRIPLE Project

Carnegie Mellon University

16 January 2004

*POPL '04*, Venice, Italy

(Revised 19 January 2004)

# Outline

- Overview
- Language
- Bidirectional Typing
- Property Types
- Contextual Typing Annotations
- Related Work
- Conclusion

# Motivation

- Conventional type systems (ML, Java, ...)
  - Decidable
  - Easy to use
  - Not so expressive
  
- Fully dependent type systems (Nuprl, ...)
  - Very expressive
  - Undecidable

# Motivation

- Conventional type systems (ML, Java, ...)
  - Decidable
  - Easy to use
  - Not so expressive
- Refined type systems
  - Easy to use
  - More expressive
  - Decidable (but not with full inference)
- Fully dependent type systems (Nuprl, ...)
  - Very expressive
  - Undecidable

# Overview

- Our work features **property types**:
  - $\delta(i)$ : Datatype with datasort and index refinement
  - $\wedge$ : Intersection:  $v : A \wedge B$  means  $v$  has type  $A$  *and* type  $B$
  - $\Pi$ : Universal index quantification (infinitary  $\wedge$ )
  - $\top$ : Greatest type (0-ary  $\wedge$ )
  - $\vee$ : Union:  $v : A \vee B$  means  $v$  has type  $A$  *or* type  $B$
  - $\Sigma$ : Existential index quantification (infinitary  $\vee$ )
  - $\perp$ : Empty type (0-ary  $\vee$ )

# Overview

- [Dunfield & Pf. '03]: A pure **type assignment** system
  - $\delta(i)$ ,  $\wedge$ ,  $\Pi$ ,  $\top$ ,  $\vee$ ,  $\Sigma$ ,  $\perp$
  - No typing annotations; **undecidable**
  - Contextual rules for indefinite types  $\vee$ ,  $\Sigma$ ,  $\perp$
  - Preservation & Progress (cbv, possible effects)
- This paper:
  - $\delta(i)$ ,  $\wedge$ ,  $\Pi$ ,  $\top$ ,  $\vee$ ,  $\Sigma$ ,  $\perp$
  - Some typing annotations; **decidable**
  - Bidirectional: *synthesis*  $e \uparrow A$ , *checking*  $e \downarrow A$
  - Bidirectional + Contextual rules = **Tridirectional**

# Datasort Refinements

- a.k.a. refinement types [Freeman+Pf. '91, Davies '97]
- Refine an algebraic datatype by a datasort  $\delta$
- Example: Lists of integers

Nil : **1**  $\rightarrow$  list

Nil : **1**  $\rightarrow$  even

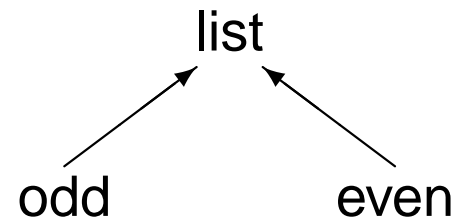
Cons : int \* list  $\rightarrow$  list

Cons : (int \* odd  $\rightarrow$  even)

$\wedge$  (int \* even  $\rightarrow$  odd)

$\wedge$  (int \* list  $\rightarrow$  list)

$\delta$  :



- Intersections essential

# Index Refinements

- Dependent types restricted to a decidable constraint domain [Xi & Pfenning '99]
- Refine an algebraic datatype by an index
- Indices drawn from any decidable constraint domain, here  $\mathcal{N}$

- Example: Lists indexed by their length

$\text{Nil} : \mathbf{1} \rightarrow \text{list}$

$\text{Nil} : \mathbf{1} \rightarrow \text{list}(0)$

$\text{Cons} : \text{int} * \text{list} \rightarrow \text{list}$

$\text{Cons} : \prod a:\mathcal{N}. \text{int} * \text{list}(a) \rightarrow \text{list}(a + 1)$

- Example:

$\text{append} : \prod a:\mathcal{N}. \prod b:\mathcal{N}. \text{list}(a) * \text{list}(b) \rightarrow \text{list}(a+b)$

- Universal quantifier  $\prod$  essential
- Existential quantifier  $\Sigma$  also essential



# Language

$A, B, C ::= \mathbf{1} \mid A \rightarrow B \mid A * B \mid \delta(i)$

$\mid A \wedge B \mid \Pi a:\gamma. A \mid \top$

$\mid A \vee B \mid \Sigma a:\gamma. A \mid \perp$

$e ::= x \mid u \mid ( ) \mid \lambda x. e \mid e_1(e_2) \mid \mathbf{fix} \ u. e$

$\mid (e_1, e_2) \mid \mathbf{fst}(e) \mid \mathbf{snd}(e)$

$\mid c(e) \mid \mathbf{case} \ e \ \mathbf{of} \ m \ s$

# cbv Semantics

Values  $v ::= x \mid () \mid \lambda x. e \mid (v_1, v_2) \mid c(v)$

Evaluation contexts  $E ::= [] \mid E(e) \mid v(E)$   
 $\mid (E, e) \mid (v, E) \mid \mathbf{fst}(E) \mid \mathbf{snd}(E)$   
 $\mid c(E) \mid \mathbf{case} E \mathbf{of} m s$

$$\frac{e' \mapsto_R e''}{E[e'] \mapsto E[e'']}$$

$(\lambda x. e) v \mapsto_R [v/x] e$        $\mathbf{fst}(v_1, v_2) \mapsto_R v_1$   
 $\mathbf{fix} u. e \mapsto_R [\mathbf{fix} u. e / u] e$        $\mathbf{snd}(v_1, v_2) \mapsto_R v_2$   
 $\mathbf{case} c(v) \mathbf{of} \dots c(x) \Rightarrow e \dots \mapsto_R [v/x] e$

# Bidirectional Typing

- Pure type assignment  $\Gamma \vdash e : A$ 
  - Undecidable with  $\wedge, \dots$
- Bidirectional typing

Synthesis  $\Gamma^+ \vdash e^+ \uparrow A^-$

Checking  $\Gamma^+ \vdash e^+ \downarrow A^+$

+ input

- output

# Bidirectional Typing

- Getting started:

$$\frac{\Gamma(x) = A}{\Gamma \vdash x : A} \text{ (var)}$$

# Bidirectional Typing

- Getting started:

$$\frac{\Gamma(x) = A}{\boxed{\Gamma \vdash x \uparrow A}} \text{ (var)}$$

# Bidirectional Typing: $\rightarrow$

- **Principle:** Introduction rules **check**, elimination rules **synthesize**

$$\frac{\Gamma, x:A \vdash e : B}{\Gamma \vdash \lambda x. e : A \rightarrow B} (\rightarrow I)$$

$$\frac{\Gamma \vdash e_1 : A \rightarrow B \quad \Gamma \vdash e_2 : A}{\Gamma \vdash e_1 e_2 : B} (\rightarrow E)$$

- $e_1$  usually a variable or another application

# Bidirectional Typing: $\rightarrow$

- **Principle:** Introduction rules **check**, elimination rules **synthesize**

$$\frac{\Gamma, x:A \vdash e : B}{\Gamma \vdash \lambda x. e \downarrow A \rightarrow B} (\rightarrow I)$$
$$\frac{\Gamma \vdash e_1 : A \rightarrow B \quad \Gamma \vdash e_2 : A}{\Gamma \vdash e_1 e_2 : B} (\rightarrow E)$$

- $e_1$  usually a variable or another application

# Bidirectional Typing: $\rightarrow$

- **Principle:** Introduction rules **check**, elimination rules **synthesize**

$$\frac{\boxed{\Gamma, x:A \vdash e \downarrow B}}{\Gamma \vdash \lambda x. e \downarrow A \rightarrow B} (\rightarrow I)$$

$$\frac{\Gamma \vdash e_1 : A \rightarrow B \quad \Gamma \vdash e_2 : A}{\Gamma \vdash e_1 e_2 : B} (\rightarrow E)$$

- $e_1$  usually a variable or another application



# Bidirectional Typing: $\rightarrow$

- **Principle:** Introduction rules **check**, elimination rules **synthesize**

$$\frac{\Gamma, x:A \vdash e \downarrow B}{\Gamma \vdash \lambda x. e \downarrow A \rightarrow B} (\rightarrow I)$$

$$\frac{\boxed{\Gamma \vdash e_1 \uparrow A \rightarrow B} \quad \Gamma \vdash e_2 : A}{\Gamma \vdash e_1 e_2 : B} (\rightarrow E)$$

- $e_1$  usually a variable or another application

# Bidirectional Typing: $\rightarrow$

- **Principle:** Introduction rules **check**, elimination rules **synthesize**

$$\frac{\Gamma, x:A \vdash e \downarrow B}{\Gamma \vdash \lambda x. e \downarrow A \rightarrow B} (\rightarrow I)$$

$$\frac{\Gamma \vdash e_1 \uparrow A \rightarrow B \quad \boxed{\Gamma \vdash e_2 \downarrow A}}{\Gamma \vdash e_1 e_2 : B} (\rightarrow E)$$

- $e_1$  usually a variable or another application

# Bidirectional Typing: $\rightarrow$

- **Principle:** Introduction rules **check**, elimination rules **synthesize**

$$\frac{\Gamma, x:A \vdash e \downarrow B}{\Gamma \vdash \lambda x. e \downarrow A \rightarrow B} (\rightarrow I)$$

$$\frac{\Gamma \vdash e_1 \uparrow A \rightarrow B \quad \Gamma \vdash e_2 \downarrow A}{\Gamma \vdash e_1 e_2 \uparrow B} (\rightarrow E)$$

- $e_1$  usually a variable or another application

# Bidirectional Typing: \*

- **Principle:** Introduction rules **check**, elimination rules **synthesize**

$$\frac{\Gamma \vdash e_1 : A \quad \Gamma \vdash e_2 : B}{\Gamma \vdash (e_1, e_2) : A * B} (*I)$$

$$\frac{\Gamma \vdash e : A * B}{\Gamma \vdash \mathbf{fst}(e) : A} (*E_1) \quad \frac{\Gamma \vdash e : A * B}{\Gamma \vdash \mathbf{snd}(e) : B} (*E_2)$$

# Bidirectional Typing: \*

- **Principle:** Introduction rules **check**, elimination rules **synthesize**

$$\frac{\Gamma \vdash e_1 : A \quad \Gamma \vdash e_2 : B}{\Gamma \vdash (e_1, e_2) \downarrow A * B} (*I)$$

$$\frac{\Gamma \vdash e : A * B}{\Gamma \vdash \mathbf{fst}(e) : A} (*E_1)$$

$$\frac{\Gamma \vdash e : A * B}{\Gamma \vdash \mathbf{snd}(e) : B} (*E_2)$$

# Bidirectional Typing: \*

- **Principle:** Introduction rules **check**, elimination rules **synthesize**

$$\frac{\boxed{\Gamma \vdash e_1 \downarrow A \quad \Gamma \vdash e_2 \downarrow B}}{\Gamma \vdash (e_1, e_2) \downarrow A * B} (*I)$$

$$\frac{\Gamma \vdash e : A * B}{\Gamma \vdash \mathbf{fst}(e) : A} (*E_1)$$

$$\frac{\Gamma \vdash e : A * B}{\Gamma \vdash \mathbf{snd}(e) : B} (*E_2)$$

# Bidirectional Typing: \*

- **Principle:** Introduction rules **check**, elimination rules **synthesize**

$$\frac{\Gamma \vdash e_1 \downarrow A \quad \Gamma \vdash e_2 \downarrow B}{\Gamma \vdash (e_1, e_2) \downarrow A * B} (*I)$$

$$\frac{\boxed{\Gamma \vdash e \uparrow A * B}}{\Gamma \vdash \mathbf{fst}(e) : A} (*E_1)$$

$$\frac{\Gamma \vdash e : A * B}{\Gamma \vdash \mathbf{snd}(e) : B} (*E_2)$$

# Bidirectional Typing: \*

- **Principle:** Introduction rules **check**, elimination rules **synthesize**

$$\frac{\Gamma \vdash e_1 \downarrow A \quad \Gamma \vdash e_2 \downarrow B}{\Gamma \vdash (e_1, e_2) \downarrow A * B} (*I)$$

$$\frac{\Gamma \vdash e \uparrow A * B}{\Gamma \vdash \mathbf{fst}(e) \uparrow A} (*E_1) \quad \frac{\Gamma \vdash e : A * B}{\Gamma \vdash \mathbf{snd}(e) : B} (*E_2)$$



# Bidirectional Typing: \*

- **Principle:** Introduction rules **check**, elimination rules **synthesize**

$$\frac{\Gamma \vdash e_1 \downarrow A \quad \Gamma \vdash e_2 \downarrow B}{\Gamma \vdash (e_1, e_2) \downarrow A * B} (*I)$$

$$\frac{\Gamma \vdash e \uparrow A * B}{\Gamma \vdash \mathbf{fst}(e) \uparrow A} (*E_1) \quad \frac{\boxed{\Gamma \vdash e \uparrow A * B}}{\Gamma \vdash \mathbf{snd}(e) : B} (*E_2)$$

# Bidirectional Typing: \*

- **Principle:** Introduction rules **check**, elimination rules **synthesize**

$$\frac{\Gamma \vdash e_1 \downarrow A \quad \Gamma \vdash e_2 \downarrow B}{\Gamma \vdash (e_1, e_2) \downarrow A * B} (*I)$$

$$\frac{\Gamma \vdash e \uparrow A * B}{\Gamma \vdash \mathbf{fst}(e) \uparrow A} (*E_1)$$

$$\frac{\Gamma \vdash e \uparrow A * B}{\Gamma \vdash \mathbf{snd}(e) \uparrow B} (*E_2)$$

# Alternative Formulations

- **Principle:** Introduction rules **check**, elimination rules **synthesize**
- Following the principle yields all *necessary* rules
- Leads to some “unreasonable” annotations

$\text{fst}(\underbrace{(x, y)})$

(An unreasonable program?)

- For convenience, could add rules such as

$$\frac{\Gamma \vdash e_1 \uparrow A \quad \Gamma \vdash e_2 \uparrow B}{\Gamma \vdash (e_1, e_2) \uparrow A * B} (*I-\uparrow)$$

# Change of Direction

- For syntax-directed rules, intro checks, elim synthesizes
- Other rules:
  - Subsumption

$$\frac{\Gamma \vdash e \uparrow A' \quad \Gamma \vdash A' \leq A}{\Gamma \vdash e \downarrow A} \text{ (sub)}$$

- Subtyping rules

$$\frac{}{\mathbf{1} \leq \mathbf{1}} \quad \frac{B_1 \leq A_1 \quad A_2 \leq B_2}{A_1 \rightarrow A_2 \leq B_1 \rightarrow B_2} \quad \frac{A_1 \leq B_1 \quad A_2 \leq B_2}{A_1 * A_2 \leq B_1 * B_2}$$

- Type annotation

$$\frac{\Gamma \vdash e \downarrow A}{\Gamma \vdash (e : A) \uparrow A} \text{ (anno)}$$

# Property Types

- $A \rightarrow B$ ,  $A * B$ , ... realized by specific syntactic forms:
  - $A \rightarrow B$  introduced by  $\lambda x. e$ , eliminated by  $e_1 e_2$
  - $A * B$  introduced by  $(e_1, e_2)$ , eliminated by **fst** and **snd**
  - $\delta(i)$  introduced by  $c(e)$ , eliminated by **case**  $e$  **of**  $m$ s
- **Property types**  $A \wedge B$  (intersection),  $A \vee B$  (union), ... not realized by particular syntactic forms

# Intersection Types

- Typing:

$$\frac{\Gamma \vdash e : A_1 \wedge A_2}{\Gamma \vdash e : A_1} (\wedge E_1) \quad \frac{\Gamma \vdash e : A_1 \wedge A_2}{\Gamma \vdash e : A_2} (\wedge E_2)$$

$$\frac{\Gamma \vdash v : A_1 \quad \Gamma \vdash v : A_2}{\Gamma \vdash v : A_1 \wedge A_2} (\wedge I)$$

- Value restriction in  $(\wedge I)$  due to Davies & Pfenning '00 (mutable references)
- Again: introduction rules check, elimination rules synthesize

# Intersection Types

- Typing:

$$\frac{\boxed{\Gamma \vdash e \uparrow A_1 \wedge A_2}}{\Gamma \vdash e : A_1} (\wedge E_1) \quad \frac{\Gamma \vdash e : A_1 \wedge A_2}{\Gamma \vdash e : A_2} (\wedge E_2)$$
$$\frac{\Gamma \vdash v : A_1 \quad \Gamma \vdash v : A_2}{\Gamma \vdash v : A_1 \wedge A_2} (\wedge I)$$

- Value restriction in  $(\wedge I)$  due to Davies & Pfenning '00 (mutable references)
- Again: introduction rules check, elimination rules synthesize

# Intersection Types

- Typing:

$$\frac{\Gamma \vdash e \uparrow A_1 \wedge A_2}{\boxed{\Gamma \vdash e \uparrow A_1}} (\wedge E_1) \quad \frac{\Gamma \vdash e : A_1 \wedge A_2}{\Gamma \vdash e : A_2} (\wedge E_2)$$
$$\frac{\Gamma \vdash v : A_1 \quad \Gamma \vdash v : A_2}{\Gamma \vdash v : A_1 \wedge A_2} (\wedge I)$$

- Value restriction in  $(\wedge I)$  due to Davies & Pfenning '00 (mutable references)
- Again: introduction rules check, elimination rules synthesize



# Intersection Types

- Typing:

$$\frac{\Gamma \vdash e \uparrow A_1 \wedge A_2}{\Gamma \vdash e \uparrow A_1} (\wedge E_1) \quad \frac{\boxed{\Gamma \vdash e \uparrow A_1 \wedge A_2}}{\Gamma \vdash e : A_2} (\wedge E_2)$$

$$\frac{\Gamma \vdash v : A_1 \quad \Gamma \vdash v : A_2}{\Gamma \vdash v : A_1 \wedge A_2} (\wedge I)$$

- Value restriction in  $(\wedge I)$  due to Davies & Pfenning '00 (mutable references)
- Again: introduction rules check, elimination rules synthesize

# Intersection Types

- Typing:

$$\frac{\Gamma \vdash e \uparrow A_1 \wedge A_2}{\Gamma \vdash e \uparrow A_1} (\wedge E_1) \quad \frac{\Gamma \vdash e \uparrow A_1 \wedge A_2}{\boxed{\Gamma \vdash e \uparrow A_2}} (\wedge E_2)$$
$$\frac{\Gamma \vdash v : A_1 \quad \Gamma \vdash v : A_2}{\Gamma \vdash v : A_1 \wedge A_2} (\wedge I)$$

- Value restriction in  $(\wedge I)$  due to Davies & Pfenning '00 (mutable references)
- Again: introduction rules check, elimination rules synthesize

# Intersection Types

- Typing:

$$\frac{\Gamma \vdash e \uparrow A_1 \wedge A_2}{\Gamma \vdash e \uparrow A_1} (\wedge E_1) \quad \frac{\Gamma \vdash e \uparrow A_1 \wedge A_2}{\Gamma \vdash e \uparrow A_2} (\wedge E_2)$$

$$\frac{\Gamma \vdash v : A_1 \quad \Gamma \vdash v : A_2}{\Gamma \vdash v \downarrow A_1 \wedge A_2} (\wedge I)$$

- Value restriction in  $(\wedge I)$  due to Davies & Pfenning '00 (mutable references)
- Again: introduction rules check, elimination rules synthesize

# Intersection Types

- Typing:

$$\frac{\Gamma \vdash e \uparrow A_1 \wedge A_2}{\Gamma \vdash e \uparrow A_1} (\wedge E_1) \quad \frac{\Gamma \vdash e \uparrow A_1 \wedge A_2}{\Gamma \vdash e \uparrow A_2} (\wedge E_2)$$

$$\frac{\boxed{\Gamma \vdash v \downarrow A_1} \quad \Gamma \vdash v : A_2}{\Gamma \vdash v \downarrow A_1 \wedge A_2} (\wedge I)$$

- Value restriction in  $(\wedge I)$  due to Davies & Pfenning '00 (mutable references)
- Again: introduction rules check, elimination rules synthesize

# Intersection Types

- Typing:

$$\frac{\Gamma \vdash e \uparrow A_1 \wedge A_2}{\Gamma \vdash e \uparrow A_1} (\wedge E_1) \quad \frac{\Gamma \vdash e \uparrow A_1 \wedge A_2}{\Gamma \vdash e \uparrow A_2} (\wedge E_2)$$

$$\frac{\Gamma \vdash v \downarrow A_1 \quad \boxed{\Gamma \vdash v \downarrow A_2}}{\Gamma \vdash v \downarrow A_1 \wedge A_2} (\wedge I)$$

- Value restriction in  $(\wedge I)$  due to Davies & Pfenning '00 (mutable references)
- Again: introduction rules check, elimination rules synthesize

# Intersection Types

- Subtyping: 
$$\frac{A \leq B_1 \quad A \leq B_2}{A \leq B_1 \wedge B_2} (\wedge R)$$
$$\frac{A_1 \leq B}{A_1 \wedge A_2 \leq B} (\wedge L_1) \quad \frac{A_2 \leq B}{A_1 \wedge A_2 \leq B} (\wedge L_2)$$

- Distributivity?

$$\frac{}{(A \rightarrow B) \wedge (A \rightarrow B') \leq A \rightarrow (B \wedge B')}$$

- Unsound with mutable references. [Davies & Pfenning '00]

# Indefinite Types: Motivation

*filter* : (int → bool) → list → list

*filter* :  $\Pi \alpha : \mathcal{N}. (\text{int} \rightarrow \text{bool}) \rightarrow \text{list}(\alpha) \rightarrow \text{list}(\_)$

# Indefinite Types: Motivation

*filter* : (int → bool) → list → list

*filter* :  $\prod a:\mathcal{N}. (\text{int} \rightarrow \text{bool}) \rightarrow \text{list}(a) \rightarrow \text{list}(\_)$

*filter* :  $\prod a:\mathcal{N}. (\text{int} \rightarrow \text{bool}) \rightarrow \text{list}(a) \rightarrow (\sum b:\mathcal{N}. \text{list}(b))$

- $\sum b:\mathcal{N}. B$  quantifies existentially over  $b$  in  $B$
- [Xi & Pfenning '99]



# Union Types

- Union type  $A \vee B$ —binary analogue of  $\Sigma$
- Subtyping dual to  $\wedge$
- Introduction rules

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash e : A \vee B} (\vee I_1) \quad \frac{\Gamma \vdash e : B}{\Gamma \vdash e : A \vee B} (\vee I_2)$$

- Elimination rule: *contextual rule*  $E[e']$

$$\frac{\Gamma \vdash e' : A \vee B \quad \begin{array}{l} \Gamma, x:A \vdash E[x] : C \\ \Gamma, x:B \vdash E[x] : C \end{array}}{\Gamma \vdash E[e'] : C} (\vee E)$$

- Several alternative elim rules are unsound [Dunfield & Pf. '03]

# Union Types

- Union type  $A \vee B$ —binary analogue of  $\Sigma$
- Subtyping dual to  $\wedge$
- Introduction rules

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash e \downarrow A \vee B} (\vee I_1) \quad \frac{\Gamma \vdash e : B}{\Gamma \vdash e : A \vee B} (\vee I_2)$$

- Elimination rule: *contextual rule*  $E[e']$

$$\frac{\Gamma \vdash e' : A \vee B \quad \begin{array}{l} \Gamma, x:A \vdash E[x] : C \\ \Gamma, x:B \vdash E[x] : C \end{array}}{\Gamma \vdash E[e'] : C} (\vee E)$$

- Several alternative elim rules are unsound [Dunfield & Pf. '03]

# Union Types

- Union type  $A \vee B$ —binary analogue of  $\Sigma$
- Subtyping dual to  $\wedge$
- Introduction rules

$$\frac{\boxed{\Gamma \vdash e \downarrow A}}{\Gamma \vdash e \downarrow A \vee B} (\vee I_1) \qquad \frac{\Gamma \vdash e : B}{\Gamma \vdash e : A \vee B} (\vee I_2)$$

- Elimination rule: *contextual rule*  $E[e']$

$$\frac{\Gamma \vdash e' : A \vee B \quad \begin{array}{l} \Gamma, x:A \vdash E[x] : C \\ \Gamma, x:B \vdash E[x] : C \end{array}}{\Gamma \vdash E[e'] : C} (\vee E)$$

- Several alternative elim rules are unsound [Dunfield & Pf. '03]

# Union Types

- Union type  $A \vee B$ —binary analogue of  $\Sigma$
- Subtyping dual to  $\wedge$
- Introduction rules

$$\frac{\Gamma \vdash e \downarrow A}{\Gamma \vdash e \downarrow A \vee B} (\vee I_1) \qquad \frac{\Gamma \vdash e \downarrow B}{\Gamma \vdash e \downarrow A \vee B} (\vee I_2)$$

- Elimination rule: *contextual rule*  $E[e']$

$$\frac{\Gamma \vdash e' : A \vee B \quad \begin{array}{l} \Gamma, x:A \vdash E[x] : C \\ \Gamma, x:B \vdash E[x] : C \end{array}}{\Gamma \vdash E[e'] : C} (\vee E)$$

- Several alternative elim rules are unsound [Dunfield & Pf. '03]

# Union Types

- Union type  $A \vee B$ —binary analogue of  $\Sigma$
- Subtyping dual to  $\wedge$
- Introduction rules

$$\frac{\Gamma \vdash e \downarrow A}{\Gamma \vdash e \downarrow A \vee B} (\vee I_1) \qquad \frac{\Gamma \vdash e \downarrow B}{\Gamma \vdash e \downarrow A \vee B} (\vee I_2)$$

- Elimination rule: *contextual rule*  $E[e']$

$$\frac{\boxed{\Gamma \vdash e' \uparrow A \vee B} \quad \begin{array}{l} \Gamma, x:A \vdash E[x] : C \\ \Gamma, x:B \vdash E[x] : C \end{array}}{\Gamma \vdash E[e'] : C} (\vee E)$$

- Several alternative elim rules are unsound [Dunfield & Pf. '03]

# Union Types

- Union type  $A \vee B$ —binary analogue of  $\Sigma$
- Subtyping dual to  $\wedge$
- Introduction rules

$$\frac{\Gamma \vdash e \downarrow A}{\Gamma \vdash e \downarrow A \vee B} (\vee I_1) \qquad \frac{\Gamma \vdash e \downarrow B}{\Gamma \vdash e \downarrow A \vee B} (\vee I_2)$$

- Elimination rule: *contextual rule*  $E[e']$

$$\frac{\Gamma \vdash e' \uparrow A \vee B \quad \begin{array}{l} \Gamma, x:A \vdash E[x] \downarrow C \\ \Gamma, x:B \vdash E[x] \downarrow C \end{array}}{\Gamma \vdash E[e'] : C} (\vee E)$$

- Several alternative elim rules are unsound [Dunfield & Pf. '03]

# Union Types

- Union type  $A \vee B$ —binary analogue of  $\Sigma$
- Subtyping dual to  $\wedge$
- Introduction rules

$$\frac{\Gamma \vdash e \downarrow A}{\Gamma \vdash e \downarrow A \vee B} (\vee I_1) \quad \frac{\Gamma \vdash e \downarrow B}{\Gamma \vdash e \downarrow A \vee B} (\vee I_2)$$

- Elimination rule: *contextual rule*  $E[e']$

$$\frac{\Gamma \vdash e' \uparrow A \vee B \quad \begin{array}{l} \Gamma, x:A \vdash E[x] \downarrow C \\ \Gamma, x:B \vdash E[x] \downarrow C \end{array}}{\Gamma \vdash E[e'] \downarrow C} (\vee E)$$

- Several alternative elim rules are unsound [Dunfield & Pf. '03]

# More Contextual Rules

- Existential index quantifier  $\Sigma a:\gamma. A$

$$\frac{\Gamma \vdash e' \uparrow \Sigma a:\gamma. A \quad \Gamma, a:\gamma, x:A \vdash E[x] \downarrow C}{\Gamma \vdash E[e'] \downarrow C} (\Sigma E)$$

- Empty type  $\perp$

$$\frac{\Gamma \vdash e' \uparrow \perp}{\Gamma \vdash E[e'] \downarrow C} (\perp E)$$

- Common structure of  $(\forall E)$ ,  $(\Sigma E)$ ,  $(\perp E)$

$$\frac{\Gamma \vdash e' \uparrow \text{indefinite type} \quad \dots n \text{ premises} \dots}{\Gamma \vdash E[e'] \downarrow C}$$



# The Tridirectional Rule

- Common structure of ( $\forall E$ ), ( $\Sigma E$ ), ( $\perp E$ )

$$\frac{\Gamma \vdash e' \uparrow \text{indefinite type} \quad \dots n \text{ premises} \dots}{\Gamma \vdash E[e'] \downarrow C}$$

- Tridirectional rule

$$\frac{\Gamma \vdash e' \uparrow A \quad \Gamma, x:A \vdash E[x] \downarrow C}{\Gamma \vdash E[e'] \downarrow C} \text{ (direct)}$$

- Not admissible due to evaluation context restriction

# Type Annotations

- Usual annotation form ( $e : A$ ) not adequate:
  - Intersection types
    - Annotate with several types ( $e : A_1, \dots, A_n$ ) [Pierce '91, Reynolds '96]
    - Typechecker guesses which type to use
  - Index variables
    - Scoping issues ( $\alpha$ -conversion fails)

# Index Variable Scoping

- $(\lambda x. (\underbrace{\lambda z. x}_{\text{Annotation}}) ()) : \prod a:\mathcal{N}. \text{list}(a) \rightarrow \text{list}(a)$

Requires  
annotation

- What annotation?
- $((\lambda z. x) : \mathbf{1} \rightarrow \text{list}(a))$ 
  - $\prod a:\mathcal{N}. \underbrace{\text{list}(a) \rightarrow \text{list}(a)}_{\text{Natural scope of } a}$
  - $\alpha$ -converting should change nothing:
  - $(\lambda z. x : \mathbf{1} \rightarrow \text{list}(a)) \cdots : \prod b:\mathcal{N}. \text{list}(b) \rightarrow \text{list}(b)$
  - But  $\text{list}(a)$  is now meaningless

# Index Variable Scoping

- Syntactic marker  $\Lambda a:\gamma. e$  for  $\Pi$  fails with intersections:

$rev : (\Pi a:\mathcal{N}. list(a) \rightarrow list(a))$  needs one  $\Lambda$   
 $\wedge ((\Sigma b:\mathcal{N}. list(b)) \rightarrow \Sigma c:\mathcal{N}. list(c))$  needs no  $\Lambda$ s

# Contextual Typing Annotations

$$(e : (\Gamma_1 \vdash A_1, \dots, \Gamma_n \vdash A_n))$$

- List of typings  $\Gamma_k \vdash A_k$
- Choose typing  $\Gamma_k \vdash A_k$  if  $\Gamma$  supports  $\Gamma_k$
- Safe under  $\alpha$ -conversion
- Example:

$$((\lambda y. \text{Cons}(42, x)) : x:\text{even} \vdash \mathbf{1} \rightarrow \text{odd}, \\ x:\text{odd} \vdash \mathbf{1} \rightarrow \text{even})$$

- If typechecking under  $\dots, x:\text{even}$ , use 1st typing
- If typechecking under  $\dots, x:\text{odd}$ , use 2nd typing
- Details and small examples in paper

# Theorems

- **Soundness:** If  $\vdash e \downarrow A$  or  $\vdash e \uparrow A$  then  $\vdash |e| : A$ .
- **Completeness:** If  $\vdash e : A$  then  $\vdash e' \downarrow A$  where  $e'$  is an annotated version of  $e$ .
- **Preservation + Progress** (cbv with effects): Follows from result for the type assignment system [Dunfield & Pf. '03].
- **Decidability of Typechecking:** Not immediate (can repeatedly apply (direct)); shown via left-rule system (inspired by [Barbanera et al.'95])

# Related Work

- [Davies '97, Davies & Pfenning '00]: Bidirectional system with  $\delta$ ,  $\wedge$
- [Xi & Pfenning '99]: Bidirectional system with  $i$ ,  $\Pi$ ,  $\Sigma$
- [Coppo et al.'81]:  $\wedge$  can characterize normal forms (termination); hence full inference undecidable
- [Pierce & Turner '00]: Local type inference
- [Reynolds '96]: FORSYTHE with  $\wedge$  (and type annotations)
- [Pierce '91]: Language with  $\wedge$ ,  $\vee$ , syntactic markers
- [Jim '95, Wells '02]: Principal typings

# Conclusion



# Summary

- Goal: express more invariants
- To express properties of algebraic datatypes:
  - Datasort refinements
  - Index refinements
- To combine properties:  $\wedge$ ,  $\vee$ ,  $\Pi$ ,  $\Sigma$
- Bidirectional typechecking  
+ Elim rules for  $\vee$ ,  $\Sigma$ ,  $\perp$  on evaluation contexts  
= **Tridirectional typechecking**
- Contextual typing annotations
- Preservation + Progress
- Decidable

# Future Work

- Let-normal (“2/3 CPS”) translation to reduce  $E[e']$  nondeterminism
  - Connections with CPS & program analysis [Sabry & Felleisen '94, Damian & Danvy '00, Palsberg & Wand '02]
- Implementation
- Parametric polymorphism (even w/o distributivity, undecidable for impredicative [Chrzęszcz '98])

# The End