Intersection Types and Computational Effects

Rowan Davies and Frank Pfenning

ICFP 2000, Montreal, Canada

September 19, 2000

What are Intersection Types?

- Introduction rule

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash M : B}{\Gamma \vdash M : A \wedge B}$$

- Elimination via subtyping

$$\frac{\Gamma \vdash M : A \qquad A \leq B}{\Gamma \vdash M : B}$$

- Intersection as a greatest lower bound

$$\overline{A \wedge B \leq A} \qquad \overline{A \wedge B \leq B}$$

$$\frac{A \leq B \qquad A \leq C}{A \leq B \wedge C}$$

Why Intersection Types?

- Admit **more** programs as well-typed.

$$\vdash (\lambda x.\, x\, x) : ((A \to B) \wedge A) \to B$$

- Admit **fewer** programs as well-typed.
  (see next slide)

- Type refinement and program invariants.

- Express more than one property of a program.

- Intersection as finitary polymorphism.

Example: Bit Strings and Natural Numbers

$$\textit{Bit Strings} \qquad \mathsf{bits} \ ::= \ \epsilon \mid \mathsf{bits}\,\mathbf{1} \mid \mathsf{bits}\,\mathbf{0}$$

$$\textit{Natural Numbers} \qquad \mathsf{nat} \ ::= \ \epsilon \mid \mathsf{pos}$$

$$\textit{Positive Numbers} \qquad \mathsf{pos} \ ::= \ \mathsf{pos}\,\mathbf{0} \mid \mathsf{nat}\,\mathbf{1}$$

- ϵ represents zero, $\mathbf{0}$ and $\mathbf{1}$ are postfix operators.

- Natural numbers have no leading $\mathbf{0}$s.

- Consider $double = \lambda x.\, x\,\mathbf{0}$.

  We have

  $$\vdash \lambda x.\, x\,\mathbf{0} \ : \ (\mathsf{bits} \rightarrow \mathsf{bits})$$
  $$\wedge\ (\mathsf{nat} \rightarrow \mathsf{bits})$$
  $$\wedge\ (\mathsf{pos} \rightarrow \mathsf{pos})$$

  But(!)

  $$\nvdash \lambda x.\, x\,\mathbf{0} \ : \ (\mathsf{nat} \rightarrow \mathsf{nat})$$

Intersections are Unsound with Effects

- Counterexample

$$\text{let}\quad x = \text{ref}\,(\epsilon\,\mathbf{1})\quad : \text{nat ref} \wedge \text{pos ref}$$
$$\text{in}$$
$$\begin{array}{ll} x := \epsilon; & \%\ \textit{use}\ x : \text{nat ref} \\ !\,x & \%\ \textit{use}\ x : \text{pos ref} \end{array}$$
$$\text{end}\quad : \text{pos}$$

  evaluates to ϵ which does not have type pos.

- Analogous counterexample with parametric polymorphism:

$$\text{let}\quad x = \text{ref}\,(\lambda y.\,y)\quad : \forall\alpha.\,(\alpha \to \alpha)\,\text{ref}$$
$$\text{in}$$
$$\begin{array}{ll} x := (\lambda y.\,\epsilon); & \%\ \textit{use}\ x : (\text{nat} \to \text{nat})\,\text{ref} \\ (!\,x)\,(\epsilon\,\mathbf{1}) & \%\ \textit{use}\ x : (\text{pos} \to \text{pos})\,\text{ref} \end{array}$$
$$\text{end}\quad : \text{pos}$$

Overview

- A Value Restriction

- Distributivity

- Progress and Type Preservation

- Bi-Directional Type Checking

- Parametric Polymorphism

- Conclusion

Subtyping

$$Types \quad A \quad ::= \quad \mathsf{bits} \mid \mathsf{nat} \mid \mathsf{pos}$$
$$\mid A_1 \rightarrow A_2 \mid A\,\mathsf{ref} \mid \mathsf{unit}$$
$$\mid A_1 \wedge A_2$$

$$\frac{}{A \leq A} \qquad \frac{A \leq B \qquad B \leq C}{A \leq C} \qquad \leq: \text{ Reflexive and transitive}$$

$$\frac{B_1 \leq A_1 \qquad A_2 \leq B_2}{A_1 \rightarrow A_2 \leq B_1 \rightarrow B_2} \qquad \rightarrow: \text{ Contra- and co-variant}$$

$$\frac{A \leq B \quad B \leq A}{A\,\mathsf{ref} \leq B\,\mathsf{ref}} \qquad \mathsf{ref}: \text{ Non-variant}$$

Subtyping and Intersections

$$\overline{\mathsf{pos} \leq \mathsf{nat}} \qquad \overline{\mathsf{nat} \leq \mathsf{bits}} \qquad\qquad \text{Data types}$$

$$\overline{A \wedge B \leq A} \qquad \overline{A \wedge B \leq B} \qquad\qquad \wedge:\ \text{Lower bound}$$

$$\frac{A \leq B \qquad A \leq C}{A \leq B \wedge C} \qquad\qquad \wedge:\ \text{Greatest lower bound}$$

$$\left[\ \overline{(A \rightarrow B) \wedge (A \rightarrow C) \leq A \rightarrow (B \wedge C)}\ \right] \qquad \text{?? (Distributivity)}$$

- Distributivity disturbs orthogonality of constructors.

- Distributivity is unsound with effects (see later).

Typing Judgment

- Language is standard call-by-value language with functions, mutable references, unit, bit strings, let and recursion.

- Use pure type assignment for typeless operational semantics.

- Later: bi-directional type-checking.

- Pragmatically: refinement restriction?

- Typing rules are standard for functions, recursion, references.

Typing Bit Strings

- Bit strings (two rules for case omitted):

$$\frac{}{\Gamma \vdash \epsilon : \mathsf{nat}}$$

$$\frac{\Gamma \vdash M : \mathsf{pos}}{\Gamma \vdash M\,\mathbf{0} : \mathsf{pos}} \qquad \frac{\Gamma \vdash M : \mathsf{bits}}{\Gamma \vdash M\,\mathbf{0} : \mathsf{bits}}$$

$$\frac{\Gamma \vdash M : \mathsf{nat}}{\Gamma \vdash M\,\mathbf{1} : \mathsf{pos}} \qquad \frac{\Gamma \vdash M : \mathsf{bits}}{\Gamma \vdash M\,\mathbf{1} : \mathsf{bits}}$$

$$\frac{\Gamma \vdash M : \mathsf{pos} \quad \Gamma, x{:}\mathsf{pos} \vdash N_0 : A \quad \Gamma, y{:}\mathsf{nat} \vdash N_1 : A}{\Gamma \vdash \mathsf{case}\, M \,\mathsf{of}\, \epsilon \Rightarrow N_e \mid x\,\mathbf{0} \Rightarrow N_0 \mid y\,\mathbf{1} \Rightarrow N_1 : A}$$

- Note: $\mathsf{case}\,(M{:}\mathsf{pos})$ does not need to check N_e.

Typing Judgment Continued

- Value restriction and subsumption.

$$\frac{\Gamma \vdash V : A \qquad \Gamma \vdash V : B}{\Gamma \vdash V : A \wedge B} \qquad\qquad \frac{\Gamma \vdash M : A \qquad A \leq B}{\Gamma \vdash M : B}$$

  where

$$\text{Values} \quad V \quad ::= \quad x \mid \lambda x.\, M \mid \epsilon \mid V\, \mathbf{0} \mid V\, \mathbf{1}$$

- Originally introduced for parametric polymorphism [Tofte'90] [Wright'95].

- Value restriction here not tied to let!

$$\frac{\Gamma \vdash M : A \qquad \Gamma, x{:}A \vdash N : B}{\Gamma \vdash \mathsf{let}\, x = M \,\mathsf{in}\, N \,\mathsf{end} : B}$$

Counterexample Revisited

$$\text{let } \; x = \mathsf{ref}\,(\epsilon\,\mathbf{1}) \quad : \mathsf{nat\ ref} \wedge \mathsf{pos\ ref}$$

$$\text{in}$$

$$x := \epsilon; \qquad\qquad \textit{\% use } x : \mathsf{nat\ ref}$$

$$!\,x \qquad\qquad\quad \textit{\% use } x : \mathsf{pos\ ref}$$

$$\text{end} \; : \mathsf{pos}$$

- No longer well typed:

$$\nvdash \mathsf{ref}\,(\epsilon\,\mathbf{1}) : \mathsf{nat\ ref} \wedge \mathsf{pos\ ref}$$

since $\mathsf{ref}\,(\epsilon\,\mathbf{1})$ is not a value.

Distributivity Revisited

- Distributivity is unsound with effects.

$$\left[\overline{(A \to B) \wedge (A \to C) \leq A \to (B \wedge C)} \right]$$

- Counterexample:

$$\vdash \lambda u.\, \mathsf{ref}\,(\epsilon\,\mathbf{1}) \qquad : \quad (\mathsf{unit} \to \mathsf{nat\ ref}) \wedge (\mathsf{unit} \to \mathsf{pos\ ref})$$

by distributivity and subsumption:

$$\vdash \lambda u.\, \mathsf{ref}\,(\epsilon\,\mathbf{1}) \qquad : \quad \mathsf{unit} \to (\mathsf{nat\ ref} \wedge \mathsf{pos\ ref})$$

$$\vdash (\lambda u.\, \mathsf{ref}\,(\epsilon\,\mathbf{1}))\,\langle\,\rangle \quad : \quad \mathsf{nat\ ref} \wedge \mathsf{pos\ ref}$$

- In a program:

$$\mathsf{let}\ \ x = (\lambda u.\, \mathsf{ref}\,(\epsilon\,\mathbf{1}))\,\langle\,\rangle \quad : \mathsf{nat\ ref} \wedge \mathsf{pos\ ref}$$

$$\mathsf{in}\ \ \ldots\ \mathsf{end} \qquad\qquad\qquad \textit{\% as on slide 5}$$

Results

- **Theorem:** Subtyping is structural.

- **Lemma:** *(Typing Inversion)* With a *store typing* Δ:

  1. If $\Delta; \cdot \vdash V : A$ and $A \leq B \to C$
     then $V = \lambda x.\, M$ and $\Delta; x{:}B \vdash M : C$.

  2. ... *(one for each type or type constructor)* ...

  Fails in the presence of distributivity!

- **Theorem:** Call-by-value reduction semantics satisfies *progress* and *type preservation*.

- **Proof:** Follows [Wright & Felleisen '94] [Harper'94], using above inductive inversion properties.
  Fails in the presence of unrestricted intersection!

Consequences

- Language has no principal types:

$$\vdash \mathsf{ref}\,(\epsilon\,\mathbf{1}) \ : \ \mathsf{bits\ ref}$$

$$\vdash \mathsf{ref}\,(\epsilon\,\mathbf{1}) \ : \ \mathsf{nat\ ref}$$

$$\vdash \mathsf{ref}\,(\epsilon\,\mathbf{1}) \ : \ \mathsf{pos\ ref}$$

but bits ref, nat ref and pos ref are unrelated and

$$\nvdash \mathsf{ref}\,(\epsilon\,\mathbf{1}) \ : \ \mathsf{bits\ ref} \wedge \mathsf{nat\ ref} \wedge \mathsf{pos\ ref}$$

Bi-Directional Type-Checking

- Simplified subtyping allows simplified bi-directional type-checking.

- Functional fragment

$$\text{Inferable} \quad I \quad ::= \quad x \mid I\,C \mid C{:}A$$

$$\text{Checkable} \quad C \quad ::= \quad I \mid \lambda x.\,C$$

- Normal forms require no type annotations.

- Two mutually recursive judgments:

$$\Gamma \vdash I \uparrow A \qquad I \text{ synthesizes } A \text{ (non-deterministically)}$$

$$\Gamma \vdash C \downarrow A \qquad C \text{ checks against } A$$

Bi-Directional Typing Rules

- Inferable

$$\frac{x{:}A \text{ in } \Gamma}{\Gamma \vdash x \uparrow A} \qquad \frac{\Gamma \vdash I \uparrow A \qquad A \Uparrow B_1 \to B_2 \qquad \Gamma \vdash C \downarrow B_1}{\Gamma \vdash I\,C \uparrow B_2}$$

$$\frac{\Gamma \vdash (C{:}A) \uparrow A}{\Gamma \vdash C \downarrow A} \qquad A \Uparrow B \quad B \text{ is a conjunct of } A$$

- Checkable (C_v a checkable value)

$$\frac{\Gamma \vdash I \uparrow A \qquad A \le B}{\Gamma \vdash I \downarrow B} \qquad \frac{\Gamma \vdash C_v \downarrow A \qquad \Gamma \vdash C_v \downarrow B}{\Gamma \vdash C_v \downarrow A \wedge B}$$

$$\frac{\Gamma, x{:}A \vdash M \downarrow B}{\Gamma \vdash \lambda x.\,M \downarrow A \to B}$$

Pragmatics

- No distributivity: sometimes more explicit types.

- Bi-directionality: sometimes lift local functions.

- Boolean constraints for efficient implementation (speculative)

parametric polymorphism	intersection polymorphism
type variable	boolean variable
unification	boolean constraint simplification

Adding Parametric Polymorphism

$$\textit{Types} \quad A \quad ::= \quad \ldots \mid \alpha \mid \forall \alpha.\, A$$

- Subtyping

$$\overline{\forall \alpha.\, A \leq [B/\alpha]A} \qquad\qquad \overline{A_1 \wedge A_2 \leq A_1} \quad \overline{A_1 \wedge A_2 \leq A_2}$$

$$\frac{A \leq B}{A \leq \forall \alpha.\, B}\, \alpha \notin \mathsf{FV}(A) \qquad\qquad \frac{A \leq B_1 \qquad A \leq B_2}{A \leq B_1 \wedge B_2}$$

- Distributivity is unsound.

$$\left[\frac{}{\forall \alpha.\, (A \to B) \leq A \to \forall \alpha.\, B}\, \alpha \notin \mathsf{FV}(A) \right]$$

Structural Subtyping (Sound & Complete)

$$\overline{A \trianglelefteq A}$$

$$\overline{\mathsf{pos} \trianglelefteq \mathsf{nat}} \qquad \overline{\mathsf{pos} \trianglelefteq \mathsf{bits}} \qquad \overline{\mathsf{nat} \trianglelefteq \mathsf{bits}}$$

$$\frac{B_1 \trianglelefteq A_1 \qquad A_2 \trianglelefteq B_2}{A_1 \to A_2 \trianglelefteq B_1 \to B_2} \qquad \frac{A \trianglelefteq B \qquad B \trianglelefteq A}{A \ \mathsf{ref} \trianglelefteq B \ \mathsf{ref}}$$

$$\frac{A_1 \trianglelefteq B^o}{A_1 \wedge A_2 \trianglelefteq B^o} \qquad \frac{A_2 \trianglelefteq B^o}{A_1 \wedge A_2 \trianglelefteq B^o} \qquad \frac{A \trianglelefteq B_1 \qquad A \trianglelefteq B_1}{A \trianglelefteq B_1 \wedge B_2}$$

$$\frac{[A'/\alpha]A \trianglelefteq B^o}{\forall \alpha. \, A \trianglelefteq B^o} \qquad \frac{A \trianglelefteq B}{A \trianglelefteq \forall \alpha. \, B} \, (\alpha \notin \mathsf{FV} A)$$

$B^o \neq \forall x. \, B_1$ and $B^o \neq B_1 \wedge B_2$

Properties of Subtyping

- With distributivity have [Mitchell'88].

- Subtyping then undecidable [Tiuryn & Urzyczyn'96] [Wells'95].

- Without distributivity have structural subtyping.

- Decidable?

- Orthogonal to other type constructors.

Value Restriction

- Introduction rule

$$\frac{\Gamma \vdash V : A}{\Gamma \vdash V : \forall \alpha.\, A} \; \alpha \notin \mathsf{FV}(\Gamma)$$

- Elimination via subtyping (unchanged)

$$\frac{\Gamma \vdash M : A \qquad A \leq B}{\Gamma \vdash M : B}$$

Unsoundness of Distributivity

- Counterexample:

$$\vdash \lambda u.\, \mathsf{ref}\,(\lambda y.\, y) \qquad : \quad \forall \alpha.\, \mathsf{unit} \to (\alpha \to \alpha)\, \mathsf{ref}$$

by distributivity and subsumption:

$$\vdash \lambda u.\, \mathsf{ref}\,(\lambda y.\, y) \qquad : \quad \mathsf{unit} \to \forall \alpha.\, (\alpha \to \alpha)\, \mathsf{ref}$$

$$\vdash (\lambda u.\, \mathsf{ref}\,(\lambda y.\, y))\,\langle\,\rangle \quad : \quad \forall \alpha.\, (\alpha \to \alpha)\, \mathsf{ref}$$

- In a program:

$$\mathsf{let} \quad x = (\lambda u.\, \mathsf{ref}\,(\lambda y.\, y))\,\langle\,\rangle \quad : \forall \alpha.\, (\alpha \to \alpha)\, \mathsf{ref}$$

$$\mathsf{in} \ \ldots \ \mathsf{end} \qquad\qquad\qquad \% \text{ as on slide 5}$$

Results

- **Lemma:** Typing inversion extends (without distributivity).

- **Theorem:** Progress and type preservation extend (with value restriction).

- New(?) view of value restriction and polymorphism.

Example

$$\text{val } inc \quad : \quad (\text{bits} \to \text{bits}) \wedge (\text{nat} \to \text{pos})$$
$$= \quad \text{fix } inc.\, \lambda n.\, \text{case } n$$
$$\text{of } \epsilon \Rightarrow \epsilon\, \mathbf{1}$$
$$\mid \; x\, \mathbf{0} \Rightarrow x\, \mathbf{1}$$
$$\mid \; x\, \mathbf{1} \Rightarrow (inc\; x)\, \mathbf{0}$$

$\vdash inc$	:	$\text{nat} \to \text{nat}$	*% by subtyping*
$\vdash inc$	:	$\text{pos} \to \text{pos}$	*% by subtyping*

$$\text{val } inc \quad \not{:} \quad \text{nat} \to \text{nat}$$
$$= \quad \text{fix } inc.\, \lambda n.\, \text{case } n$$
$$\text{of } \epsilon \Rightarrow \epsilon\, \mathbf{1}$$
$$\mid \; x\, \mathbf{0} \Rightarrow x\, \mathbf{1}$$
$$\mid \; x\, \mathbf{1} \Rightarrow (inc\; x)\, \mathbf{0} \qquad \textit{% missing information}$$

Example with Mutable References

$$\begin{aligned}
\text{val } count' \quad : \quad & (\text{nat ref} \rightarrow (\text{unit} \rightarrow \text{nat})) \wedge \\
& (\text{pos ref} \rightarrow (\text{unit} \rightarrow \text{pos})) \\
= \quad & \lambda c.\, \lambda x. \\
& \quad \text{let } y = \,!\, c \\
& \quad \text{in } c := inc\ y;\ y \text{ end}
\end{aligned}$$

$$\begin{aligned}
\text{val } count \quad : \quad & (\text{nat} \rightarrow (\text{unit} \rightarrow \text{nat})) \wedge \\
& (\text{pos} \rightarrow (\text{unit} \rightarrow \text{pos})) \\
= \quad & \lambda n.\ count'\ (\text{ref } n)
\end{aligned}$$

Other Examples

- More programs

$$
\begin{aligned}
\text{val } plus \quad &: \quad (\mathsf{nat} \to \mathsf{nat} \to \mathsf{nat}) \land \\
&\qquad (\mathsf{pos} \to \mathsf{nat} \to \mathsf{pos}) \land \\
&\qquad (\mathsf{nat} \to \mathsf{pos} \to \mathsf{nat}) \\[4pt]
\text{val } double \quad &: \quad (\mathsf{nat} \to \mathsf{nat}) \land (\mathsf{pos} \to \mathsf{pos}) \\[4pt]
\text{val } stdize \quad &: \quad \mathsf{bits} \to \mathsf{nat} \\[4pt]
\text{val } \omega \quad &: \quad \forall \alpha. \forall \beta. \, ((\alpha \to \beta) \land \alpha) \to \beta \\
&= \quad \lambda x. \, x \, x
\end{aligned}
$$

- More types

$$
\begin{aligned}
\mathsf{zero} \quad &::= \quad \epsilon \\[4pt]
\mathsf{even} \quad &::= \quad \epsilon \mid \mathsf{pos}\,\mathbf{0} \\[4pt]
\mathsf{odd} \quad &::= \quad \mathsf{nat}\,\mathbf{1}
\end{aligned}
$$

Related Work

- Intersection types (many)

- Forsythe [Reynolds'88] [Reynolds'96]

- Intersections and explicit polymorphism [Pierce'91] [Pierce'97]

- Refinement types [Freeman & Pf'91] [Freeman'94] [Davies'97]

- Intersection types and program analysis (many)

- Soft types (many)

- Local type inference [Pierce & Turner'97]

Future Work

- Complete implementation under refinement restriction.

- Local type inference with intersections and parametric polymorphism?

- Valuability instead of values? [Harper & Stone'00]

- Pure and impure function spaces?

- Decidability of subtyping?

Summary

- Noted unsoundness of intersection types with effects.

- Restricted intersection introduction to values.

- Simplified subtyping by removing distributivity.

- Proved type preservation and progress.

- Checked types bi-directionally.

- Added parametric polymorphism.

- Extended type preservation and progress.