Annotations for intersection typechecking

Joshua Dunfield
MPI-SWS

Kaiserslautern and Saarbrücken, Germany

29 June 2012
ITRS ’12, Dubrovnik

Outline

- **Overview**
- Elaborating unrestricted $\wedge$
- Annotations: overview
- Contextual typing annotations
- New annotations
- Summary

Context

- This paper came out of a recent project:
  - Encode language features using intersection and union
  - Elaborate away intersections and unions
- But ultimately applicable to older work
  on type refinements, $\wedge$, $\vee$

Background

- $\wedge$ (+ $\vee$) with refinement restriction
  (Freeman & Pfenning, Davies & Pfenning,
  Dunfield & Pfenning)
  - Use types to express (more) precise properties
  - Contextual typing annotations

Background

- $\wedge$ ($+ \vee$) with refinement restriction
  (Freeman & Pfenning, Davies & Pfenning,
  Dunfield & Pfenning)
  - Use types to express (more) precise properties
  - Contextual typing annotations
- $\wedge + \vee$ **without** refinement restriction
  (ICFP 2012)
  - Use types to encode language features
  - Merges lead to new type annotation forms

Background

- $\wedge$ ($+ \vee$) with refinement restriction
  (Freeman & Pfenning, Davies & Pfenning,
  Dunfield & Pfenning)
  - Use types to express (more) precise properties
  - Contextual typing annotations
- $\wedge + \vee$ **without** refinement restriction
  (ICFP 2012)
  - Use types to encode language features
  - Merges lead to new type annotation forms
    (this paper)

Refinement restriction:

- $A \wedge B$ is a well-formed type
  only if A and B refine the same base type

$(\mathsf{even} \to \mathsf{even}) \wedge (\mathsf{odd} \to \mathsf{odd})$	✓	both refine $\mathsf{bits} \to \mathsf{bits}$
$\mathsf{even} \wedge \mathsf{odd}$	✓	both refine bits
$\mathsf{int} \wedge (\mathsf{int} \to \mathsf{int})$	✗	no common base type

- With this restriction:
  - More is decidable
  - Compilation can ignore refinements and $\wedge$:
    compilation is independent of refined typechecking

Some restricted systems

- $\wedge$ + refinement types
  (Freeman & Pfenning, Davies & Pfenning)
- $\wedge$ + $\vee$ + refinement types + indexed types
  (Dunfield & Pfenning)

Some restricted systems

- $\wedge$ + refinement types
  (Freeman & Pfenning, Davies & Pfenning)
- $\wedge$ + $\vee$ + refinement types + indexed types
  (Dunfield & Pfenning)
  - But only implicitly enforced: can't compile programs that don't follow the restriction

Without the restriction:

- Forsythe-style encodings,
  e.g. records as $\wedge$ of one-field records
- Operator overloading
- Heterogeneous containers (via $\vee$)
- Entangles typechecking with compilation

Without the restriction:

- Forsythe-style encodings,
  e.g. records as $\wedge$ of one-field records
- Operator overloading
- Heterogeneous containers (via $\vee$)
- Entangles typechecking with compilation
  . . . but that's just elaboration!

Without the restriction:

- Forsythe-style encodings,
  e.g. records as $\wedge$ of one-field records
- Operator overloading
- Heterogeneous containers (via $\vee$)
- Entangles typechecking with compilation
  . . . but that's just elaboration!

Next section: Elaborating $\wedge$ (ICFP 2012).

Not the focus of this paper, but provides context.

- ✓ Overview
- ☞ **Elaborating unrestricted $\wedge$**
  - Annotations: overview
  - Contextual typing annotations
  - New annotations
  - Summary

Elaboration for $\wedge$, $\vee$

- Elaborate $\wedge$ to product and $\vee$ to disjoint sum

$$\underbrace{\mathsf{int} \wedge (\mathsf{int} \to \mathsf{int})}_{\text{type in source program}} \longrightarrow \underbrace{\mathsf{int} * (\mathsf{int} \to \mathsf{int})}_{\text{elaborated type in target program}}$$

- Old idea (Pierce, or earlier?), but never fully worked out
- Implicit $\wedge$-elimination becomes explicit $*$-elimination

$$\frac{e : A_1 \wedge A_2 \hookrightarrow M}{e : A_1 \hookrightarrow \mathbf{proj}_1\ M}\ \wedge E_1 \qquad \frac{M : A_1 * A_2}{(\mathbf{proj}_1\ M) : A_1}\ *E_1$$

Elaboration for $\wedge$, $\vee$: Merge

- We can form the type $\mathsf{int} \wedge (\mathsf{int} \to \mathsf{int})$, but is it inhabited?

Elaboration for $\wedge$, $\vee$: Merge

- We can form the type $\mathsf{int} \wedge (\mathsf{int} \to \mathsf{int})$, but is it inhabited?
- Depends on the term language!

Elaboration for $\wedge$, $\vee$: Merge

- We can form the type $\mathsf{int} \wedge (\mathsf{int} \to \mathsf{int})$, but is it inhabited?
- Depends on the term language!
- For us, the answer will be **yes**, but only because of a **merge construct**

$$e_1 \,,, e_2$$

Elaboration for $\wedge$, $\vee$: Merge

- We can form the type $\mathsf{int} \wedge (\mathsf{int} \to \mathsf{int})$,
  but is it inhabited?
- Depends on the term language!
- For us, the answer will be **yes**,
  but only because of a **merge construct**

$$e_1 \,,, e_2$$

- Above type inhabited: $0 \,,, (\lambda x.\, x + 3)$

Elaboration for $\wedge$, $\vee$: Merge

- We can form the type $\mathsf{int} \wedge (\mathsf{int} \to \mathsf{int})$,
  but is it inhabited?
- Depends on the term language!
- For us, the answer will be **yes**,
  but only because of a **merge construct**

$$e_1 \,,, e_2$$

- Above type inhabited: $0 \,,, (\lambda x.\, x + 3)$
- Generalization of the merge construct in Forsythe
  (Reynolds 1988, 1996)

Merge rule

$$\frac{\Gamma \vdash e_k : A}{\Gamma \vdash e_1 \,,, e_2 : A}\ (\exists k \in \{1, 2\})$$

Merge rule

$$\frac{\Gamma \vdash e_k : A}{\Gamma \vdash e_1 \,,, e_2 : A}\ (\exists k \in \{1,2\})$$

Example:

$$\frac{\dfrac{\cdot \vdash 0 : \mathsf{int}}{\cdot \vdash 0 \,,, (\lambda x.\, x+3) : \mathsf{int}} \qquad \dfrac{\cdot \vdash \lambda x.\, x+3 : \mathsf{int} \to \mathsf{int}}{\cdot \vdash 0 \,,, (\lambda x.\, x+3) : \mathsf{int} \to \mathsf{int}}}{\cdot \vdash 0 \,,, (\lambda x.\, x+3) \ :\ \mathsf{int} \wedge (\mathsf{int} \to \mathsf{int})}\ \wedge\mathrm{I}$$

Merge rule

$$\frac{\Gamma \vdash e_k : A}{\Gamma \vdash e_1 \,,, e_2 : A}\ (\exists k \in \{1,2\})$$

Example:

$$\frac{\dfrac{\cdot \vdash 0 : \mathsf{int}}{\cdot \vdash 0 \,,, (\lambda x.\, x+3) : \mathsf{int}} \qquad \dfrac{\cdot \vdash \lambda x.\, x+3 : \mathsf{int} \to \mathsf{int}}{\cdot \vdash 0 \,,, (\lambda x.\, x+3) : \mathsf{int} \to \mathsf{int}}}{\cdot \vdash 0 \,,, (\lambda x.\, x+3) \ :\ \mathsf{int} \wedge (\mathsf{int} \to \mathsf{int})}\ \wedge\mathrm{I}$$

- Order irrelevant: $(\lambda x.\, x+3) \,,, 0$ also OK
- Not an introduction form for $\wedge$

Summary of elaboration and merge

$$\frac{\Gamma \vdash e_k : A}{\Gamma \vdash e_1 \,,, e_2 : A}\ (\exists k \in \{1,2\})$$

- Merge $e_1 \,,, e_2$:
  a way to introduce unrestricted $\wedge$s
- For more, including operational semantics,
  see the ICFP paper `http://arxiv.org/abs/1206.5386`
- **Key point for this talk:**

  The merge lets you write **two terms** in one position

- ✓ Overview
- ✓ Elaborating unrestricted $\wedge$
- ☞ **Annotations: overview**
  - Contextual typing annotations
  - New annotations
  - Summary

Annotation forms

- In languages based on ordinary λ-calculus:

$$e : A \qquad \text{or} \qquad \lambda x : A.\, e$$

- Without intersection types:
  - (Sub)terms $\Longleftrightarrow$ (sub)derivations
  - $\therefore$ Annotations $\Longleftrightarrow$ subderivations

Annotation forms

- In languages with this introduction rule:

$$\frac{\begin{array}{cc}\mathcal{D}_1 & \mathcal{D}_2 \\ e : A_1 & e : A_2\end{array}}{e : A_1 \wedge A_2}\ \wedge\mathrm{I}$$

  - One subterm $\Longrightarrow$ multiple subderivations

$\therefore$ One annotation $\Longrightarrow$ multiple subderivations

Why we need multiple annotations

- Assume a type bits of bitstrings, refined by odd and even, denoting bitstrings of odd and even parity.
  Appending a one, $x \cdot 1$, should flip the parity:

$$(\lambda x.\, x \cdot 1) \;:\; (\mathsf{odd} \to \mathsf{even}) \wedge (\mathsf{even} \to \mathsf{odd})$$

- Different assumptions in branches:

$$\dfrac{\dfrac{x : \mathsf{odd} \vdash x \cdot 1 : \mathsf{even}}{\cdot \vdash (\lambda x.\, x \cdot 1) : (\mathsf{odd} \to \mathsf{even})} \qquad \dfrac{x : \mathsf{even} \vdash x \cdot 1 : \mathsf{odd}}{\cdot \vdash (\lambda x.\, x \cdot 1) : (\mathsf{even} \to \mathsf{odd})}}{\cdot \vdash (\lambda x.\, x \cdot 1) \;:\; (\mathsf{odd} \to \mathsf{even}) \wedge (\mathsf{even} \to \mathsf{odd})}\ \wedge\mathrm{I}$$

- No single type for the use of x in $x \cdot 1$

Design space

$e : A$

$\lambda x : A.\, e$

$\lambda x : \mathsf{odd}|\mathsf{even}.\, x \cdot 1$ (Reynolds 1988, Pierce 1991)

$(x : \mathsf{odd}, \mathsf{even}) \cdot 1$ (Davies 2005)

Design space

$e : A$

$\lambda x : A.\, e$

$\lambda x : \mathsf{odd}|\mathsf{even}.\, x \cdot 1$ (Reynolds 1988, Pierce 1991)

$(x : \mathsf{odd}, \mathsf{even}) \cdot 1$ (Davies 2005)

$\big(x : (x{:}\mathsf{odd} \vdash \mathsf{odd},\ x{:}\mathsf{even} \vdash \mathsf{even})\big) \cdot 1$ "Contextual typing annotation" (Dunfield & Pfenning 2004)

Design space

$e : A$

$\lambda x : A.\, e$

$\lambda x : \mathsf{odd}|\mathsf{even}.\, x \cdot 1$ (Reynolds 1988, Pierce 1991)

$(x : \mathsf{odd}, \mathsf{even}) \cdot 1$ (Davies 2005)

$\big(x : (x{:}\mathsf{odd} \vdash \mathsf{odd},\ x{:}\mathsf{even} \vdash \mathsf{even})\big) \cdot 1$ "Contextual typing annotation" (Dunfield & Pfenning 2004)

$\big((x \cdot 1) : (x{:}\mathsf{odd} \vdash \mathsf{even},\ x{:}\mathsf{even} \vdash \mathsf{odd})\big)$ . . . highly general

- ✓ Overview
- ✓ Elaborating unrestricted $\wedge$
- ✓ Annotations: overview
- ☞ **Contextual typing annotations**
  - New annotations
  - Summary

Contextual Typing Annotations

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

- List of typings $\Gamma_k \vdash A_k$
- Choose typing $\Gamma_k \vdash A_k$ if Γ **supports** Γ_k
- Example:
$$\big((x \cdot 1) : (x{:}\mathsf{odd} \vdash \mathsf{even}, \ x{:}\mathsf{even} \vdash \mathsf{odd})\big)$$
  - If checking under $\ldots, x{:}\mathsf{odd}$, use 1st typing
  - If checking under $\ldots, x{:}\mathsf{even}$, use 2nd typing
- Supports subtyping: suppose $\mathsf{even} \leq \mathsf{bits}$ (and $\mathsf{even} \not\leq \mathsf{list}$):
$$\big((\mathrm{id}\ x) : (x{:}\mathsf{list} \vdash \mathsf{list}, \ x{:}\mathsf{bits} \vdash \mathsf{bits})\big)$$
  - If checking under $\ldots, x{:}\mathsf{even}$: use 2nd typing

Rules for contextual typing annotations

$$\frac{}{(\cdot \vdash A) \lesssim (\Gamma \vdash A)} \lesssim\text{-empty}$$

$$\frac{\Gamma \vdash \Gamma(x) \leq B_0 \qquad (\Gamma_0 \vdash A_0) \lesssim (\Gamma \vdash A)}{(x{:}B_0, \Gamma_0 \vdash A_0) \lesssim (\Gamma \vdash A)} \lesssim\text{-pvar}$$

$$\frac{(\Gamma_0 \vdash A_0) \lesssim (\Gamma \vdash A) \qquad \Gamma \vdash e : A}{\Gamma \vdash \big(e : \ldots, (\Gamma_0 \vdash A_0), \ldots\big) : A} \text{ctx-anno}$$

"Contextual sub-typing" $(\Gamma_0 \vdash A_0) \lesssim (\Gamma \vdash A)$ if:

- Γ_0 is supported by Γ, and
- $A_0 = A$ (for now)

Mechanisms of contextual typing annotations

$$\frac{\overbrace{(\Gamma_0 \vdash A_0) \lesssim (\Gamma \vdash A)} \qquad \overbrace{\Gamma \vdash e : A}}{\Gamma \vdash \left(e : \underbrace{\ldots, (\Gamma_0 \vdash A_0), \ldots}\right) : A}\ \text{ctx-anno}$$

one context supporting another

ordinary annotation

multiplicity

- **Contextuality**: Types $A_0, \ldots$ guarded by contexts $\Gamma_0, \ldots$
- **Ordinary annotation**: Checking against annotated type
- **Multiplicity**: Several annotations on one term

- ✓ Overview
- ✓ Elaborating unrestricted $\wedge$
- ✓ Annotations: overview
- ✓ Contextual typing annotations
- ☞ **New annotations**
- Summary

Multiplicity

$$\frac{\overbrace{(\Gamma_0 \vdash A_0) \lesssim (\Gamma \vdash A)} \qquad \overbrace{\Gamma \vdash e : A}}{\Gamma \vdash \big(e : \underbrace{\ldots, (\Gamma_0 \vdash A_0), \ldots}_{\textbf{multiplicity}}\big) : A}\ \text{ctx-anno}$$

one context supporting another

ordinary annotation

- **Multiplicity**: Several annotations on one term
- **Merge** $e_1 \mathbin{„} e_2$: Several terms in one position

Therefore:

- **Multiplicity**: Merge with different annotations

$$\big(e : \ldots\big) \mathbin{„} \ldots \mathbin{„} \big(e : (\Gamma_0 \vdash A_0)\big) \mathbin{„} \ldots$$

Multiplicity

$$\frac{\overbrace{(\Gamma_0 \vdash A_0) \lesssim (\Gamma \vdash A)} \quad \overbrace{\Gamma \vdash e : A}}{\Gamma \vdash \big(e : \underbrace{\ldots, (\Gamma_0 \vdash A_0), \ldots}_{\textbf{multiplicity}}\big) : A}\ \text{ctx-anno}$$

one context supporting another

ordinary annotation

- **Multiplicity**: Several annotations
- **Merge** $e_1 \,,, e_2$: Several terms

Concretely:

$$\frac{\Gamma \vdash (x : \mathsf{even}) : \mathsf{even}}{\Gamma \vdash \underbrace{(x : \mathsf{even})}_{\text{ordinary annotation}} ,, (x : \mathsf{odd}) : \mathsf{even}}$$

Ordinary annotation

$$\frac{\overbrace{(\Gamma_0 \vdash A_0) \lesssim (\Gamma \vdash A)} \qquad \overbrace{\Gamma \vdash e : A}}{\Gamma \vdash \big(e : \underbrace{\ldots, (\Gamma_0 \vdash A_0), \ldots}_{\text{multiplicity}}\big) : A}\ \text{ctx-anno}$$

one context supporting another

ordinary annotation

- **Ordinary annotation rule**

$$\frac{\Gamma \vdash e : A}{\Gamma \vdash (e : A) : A}\ \text{right-anno}$$

Concretely:

$$\frac{\dfrac{\Gamma \vdash x : \mathsf{even}}{\Gamma \vdash (x : \mathsf{even}) : \mathsf{even}}\ \text{right-anno}}{\Gamma \vdash \underbrace{(x : \mathsf{even})}_{\text{ordinary annotation}} ,, (x : \mathsf{odd}) \ : \ \mathsf{even}}$$

Contextuality

$$\frac{\overbrace{(\Gamma_0 \vdash A_0) \lesssim (\Gamma \vdash A)} \qquad \overbrace{\Gamma \vdash e : A}}{\Gamma \vdash \big(e : \underbrace{\ldots, (\Gamma_0 \vdash A_0), \ldots}_{\text{multiplicity}}\big) : A}\ \text{ctx-anno}$$

one context supporting another

ordinary annotation

- **Contextuality**

$$\frac{\Gamma \vdash x : A \qquad \Gamma \vdash e : B}{\Gamma \vdash \big(\underbrace{x{:}A >\!:\!> e}_{\text{“left-hand” annotation}}\big) : B}\ \text{left-anno}$$

Example:

$$\dfrac{\dfrac{\Gamma \vdash x \cdot 1 : \mathsf{even}}{\Gamma \vdash (x \cdot 1 : \mathsf{even}) : \mathsf{even}}\ \text{right-anno}}{\Gamma \vdash \underbrace{x{:}\mathsf{odd}}_{\text{left-hand annotation}} >\!:\!> \underbrace{(x \cdot 1 : \mathsf{even})}_{\text{ordinary annotation}} : \mathsf{even}}\ \text{left-anno}$$

A theorem

Theorem (Encoding Contextual Typing Annotations)
If $\Gamma \vdash e : A$ with rule ctx-anno available
then $\Gamma \vdash \mathsf{trans}(e') : A$ without applying rule ctx-anno.

A theorem

Theorem (Encoding Contextual Typing Annotations)
If $\Gamma \vdash e : A$ with rule ctx-anno available
then $\Gamma \vdash \mathsf{trans}(e') : A$ without applying rule ctx-anno.

Therefore:

Ordinary annotations $x : A$

$+$ Left-hand annotations $x{:}A >:> e$

$+$ Merges $e_1 \,,, e_2$

$\geq$ Contextual typing annotations

Contextual Modal Types (Nanevski et al.)

- $A\,[\Psi]$ represents data of type A closed under context Ψ
- Annotation with a contextual type is like ordinary annotation and left-hand annotation:

$$(x \cdot 1) \;:\; \mathsf{even}\,[x{:}\mathsf{odd}]$$

Contextual Modal Types (Nanevski et al.)

- $A\,[\Psi]$ represents data of type A closed under context Ψ
- Annotation with a contextual type is like ordinary annotation and left-hand annotation:

$$(x \cdot 1) \;:\; \mathsf{even}\,[x{:}\mathsf{odd}]$$

- Unrestricted intersection gives us multiplicity:

$$\lambda x.\ \textbf{let}\ r = (y \cdot 1) : \mathsf{even}\,[y : \mathsf{odd}] \wedge \mathsf{odd}\,[y : \mathsf{even}]\ \textbf{in}\ r\,[x/y]$$

- Much more powerful than needed for annotation

Which is better?

- A (type system/typechecker/compiler) “engineering” question
- It depends on what you have already
- Given contextual types and unrestricted $\wedge$, use them
- Given merge and unrestricted $\wedge$, add left-hand annotations

“For more, see the paper”

- Typechecking is actually **bidirectional** (Section 3.1)
- The approach can be extended to indexed types with index variables (Section 4)
- For more on merges and elaboration, see ICFP 2012

- ✓ Overview
- ✓ Elaborating unrestricted $\wedge$
- ✓ Annotations: overview
- ✓ Contextual typing annotations
- ✓ New annotations
- ☞ **Summary**

Summary

- Exploring the design space of annotations in intersection type systems
- New(ish) mechanisms: merges and left-hand annotations
- Some idea of relative power:

$$\text{Contextual types}$$
$$\geq$$
$$\text{Ordinary annotations} + \text{Left-hand annotations} + \text{Merges}$$
$$\geq$$
$$\text{Contextual typing annotations}$$

Thank you

- And thanks to
  - the ITRS reviewers
  - Neelakantan R. Krishnaswami
  - Frank Pfenning

- Further reading:
  `http://www.cs.cmu.edu/~joshuad/papers/sectanno/`
  `http://www.cs.cmu.edu/~joshuad/papers/intcomp/`