Freedom vs Discipline.

"Polymorphism"

Schemas (Curry Hindley Milner)

Parameters (Reynolds, Russell, Rod)

types or algebras (implementation)

Inheritance (Simula, Smalltalk, Amber)

A semantics of Multiple Inheritance

Luca Cardelli

2. Outline of talk

Part 1

Informal examples

Informal typing rules

Part 2

Formal semantics

Type inference systems

Typechecking algorithms

Syntax (Typed monomorphic λ-calculus + records and variants)

Part 2

$e ::=$	expressions
$x \mid$	identifiers
$b \mid$	constants
$\underline{\text{if}}\ e\ \underline{\text{then}}\ e\ \underline{\text{else}}\ e \mid$	conditionals
$(a_i = e_i) \mid e.a \mid$	records
$[a = e] \mid e\ \underline{\text{is}}\ a \mid e\ \underline{\text{as}}\ a \mid$	variants
$\lambda x{:}\tau.\ e \mid e\ e \mid$	functions
$\underline{\text{rec}}\ x{:}\tau.\ e \mid$	recursive data
$e : \tau \mid$	type specs
(e)	

$\tau ::=$	type expressions
$z \mid$	type constants
$(a_i : \tau_i) \mid$	record types
$[a_i : \tau_i] \mid$	variant types
$\tau \to \tau \mid$	function types
(τ)	

Semantics of Multiple Inheritance

Expressions

$e ::= x \mid$

$\quad true \mid false \mid if\ e\ then\ e\ else\ e \mid$

$\quad 0 \mid succ \mid = \mid$

$\quad (\!| a_1 = e_1 ; \ldots ; a_n = e_n |\!) \mid e.a \mid$

$\quad [\![a = e]\!] \mid e\ is\ a \mid e\ as\ a \mid$

$\quad e\ e \mid$

$\quad \lambda x.\ e$

$a ::= \ldots$ (labels)

Semantic Domain

$$V \simeq \mathbb{T} + \mathbb{N} + (L \to V) + (L \times V) + (V \to V) + W$$

$\mathbb{T}$ flat domain of booleans

$\mathbb{N}$ flat domain of natural numbers

L flat domain of labels

$L \to V$ domain of records

$L \times V$ domain of variants

$V \to V$ domain of functions

$W = \{w\}$ run-time error

Semantics of expressions

$wrong = w \text{ in } V$

$v_i = \mathcal{E}[\![e_i]\!]\rho$

$\mathcal{E}[\![x]\!]\rho = \rho[\![x]\!]$

$\mathcal{E}[\![c]\!]\rho = \mathcal{C}[\![c]\!]$

$\mathcal{E}[\![\text{if } e_1 \text{ then } e_2 \text{ else } e_3]\!]\rho =$

$v_1 \in T \Rightarrow ((v_1 \dagger T) = true \Rightarrow v_2 \mid v_3) \mid wrong$

$\mathcal{E}[\![(a_1 = e_1, \ldots, a_n = e_n)]\!]\rho =$

$(\lambda a.\ a = a_1 \Rightarrow v_1 \mid \ldots \mid a = a_n \Rightarrow v_n \mid wrong) \text{ in } V$

$\mathcal{E}[\![e.a]\!]\rho = v \in (L \to V) \Rightarrow (v \dagger L \to V)(a) \mid wrong$

$\mathcal{E}[\![[a = e]]\!]\rho = \langle a, v \rangle \text{ in } V$

$\mathcal{E}[\![e \text{ is } a]\!]\rho = v \in (L \times V) \Rightarrow (\text{left}(v \dagger L \times V) = a) \text{ in } V \mid wrong$

$\mathcal{E}[\![e \text{ as } a]\!]\rho = v \in (L \times V) \Rightarrow ((\text{left}(v \dagger L \times V) = a) \Rightarrow \text{right}(v \dagger L \times V) \mid \bot_V) \mid wrong$

$\mathcal{E}[\![e_1\ e_2]\!]\rho = v_2 \in W \Rightarrow wrong \mid v_1 \in (V \to V) \Rightarrow (v_1 \dagger V \to V)(v_2) \mid wrong$

$\mathcal{E}[\![\lambda x.\ e]\!]\rho = (\lambda x.\ \mathcal{E}[\![e]\!]\rho[x/[\![x]\!]]) \text{ in } V$

Type Operators

$$D \boxed{\to} E = \{ f \in V \to V \mid f(D) \subseteq E \} \text{ in } V$$

$$(a : D) = \{ r \in L \to V \mid r(a) \in D \} \text{ in } V$$

$$[a : D] = \{ \langle u, v \rangle \in L \times V \mid v \in D \} \text{ in } V$$

$$(a_1 : D_1, \ldots, a_n : D_n) = (a_1 : D_1) \cap \ldots \cap (a_n : D_n)$$

$$[a_1 : D_1, \ldots, a_n : D_n] = [a_1 : D_1] \cup \ldots \cup [a_n : D_n]$$

$() = (L \to V) \text{ in } V$

$[] = \{\} \text{ in } V$

Examples:

$int \boxed{\to} int \ni \{ 3 \mapsto 4,\ 5 \mapsto 6,\ v \mapsto \bot \text{ o.w.} \}$

$\ni \{ true \mapsto 4,\ false \mapsto true,\ v \mapsto \bot \text{ o.w.} \}$

$\not\ni \{ 3 \mapsto true,\ v \mapsto \bot \text{ o.w.} \}$

$(a : int) \ni \{ a \mapsto 3,\ b \mapsto w \text{ o.w.} \}$

$\ni \{ a \mapsto 3,\ b \mapsto false,\ c \mapsto w \text{ o.w.} \}$

$\not\ni \{ a \mapsto true,\ b \mapsto w \text{ o.w.} \}$

$[a : int] \ni \langle a, 3 \rangle$

$\not\ni \langle b, 3 \rangle \quad (b \neq a)$

$\not\ni \langle a, false \rangle$

Type Inference of Multiple Inheritance

Type Terms

τ_i are type constants

α, β are type variables

σ, τ are type terms

μ, ν are monotypes (type terms with no type vars)

τ_i is a term

α is a term

$(\!|a_i : \sigma_i|\!)$ is a term if σ_i are

$[\![a_i : \sigma_i]\!]$ is a term if σ_i are

$\sigma \to \tau$ is a term if σ, τ are

Type Assignment

$v : \tau_i$ iff $v = \bot_V$ or $v \in \beta_i$

$v : (\!|a_i : \mu_i|\!)$ iff $v = \bot_V$ or $v \in (L \to V)$ and $\forall i.\ (v \restriction L \to V) a_i : \mu_i$

$v : [\![a_i : \mu_i]\!]$ iff $v = \bot_V$ or $v \in (L \times V)$ and $\exists i.\ (v \restriction L \times V) = \langle a_i, u \rangle$ with $u = \mu_i$

$v : \mu \to \nu$ iff $v = \bot_V$ or $v \in (V \to V)$ and $\forall u.\ u : \mu \Rightarrow (v \restriction V \to V)\, u : \nu$

$v : \sigma \to \tau$ iff $\forall \mu \to \nu \leq \sigma \to \tau.\quad v : \mu \to \nu$

Prop $v : d \iff v \in d$

(relate type assignment to type operator semantics)

Type Inference

$$\vdash c_{ij} : \tau_i$$

$$A \vdash x : A(x)$$

$$\frac{A \vdash e_1 : \text{bool} \quad A \vdash e_2 : \sigma \quad A \vdash e_3 : \sigma}{A \vdash (\text{if } e_1 \text{ then } e_2 \text{ else } e_3) : \sigma}$$

$$\frac{A \vdash e_1 : \sigma \to \tau \quad A \vdash e_2 : \sigma}{A \vdash e_1\, e_2 : \tau}$$

$$\frac{A[\sigma/x] \vdash e : \tau}{A \vdash (\lambda x.e) : \sigma \to \tau}$$

$$\frac{A \vdash e_i : \sigma_i}{A \vdash (\![a_i = e_i]\!) : \tau} \quad \tau \subseteq (\![a_i : e_i]\!)$$

$$\frac{A \vdash e : (\![a_i : \sigma_i]\!)}{A \vdash e.a : \tau} \quad a : \tau \in (\![a_i : \sigma_i]\!)$$

$$\frac{A \vdash e : \sigma}{A \vdash [\![a = e]\!] : [\![a_i : \tau_i]\!]} \quad a : \sigma \in [\![a_i : \tau_i]\!]$$

$$\frac{A \vdash e : [\![a_i : \sigma_i]\!]}{A \vdash e \text{ is } a : \text{bool}} \quad a : \tau \in [\![a_i : \sigma_i]\!]$$

$$\frac{A \vdash e : [\![a_i : \sigma_i]\!]}{A \vdash e \text{ as } a : \tau} \quad a : \tau \in [\![a_i : \sigma_i]\!]$$

Semantic soundness

If A agrees with ρ, then:

$$A \vdash e : \tau \Rightarrow \mathcal{E}[\![e]\!]\rho \in \tau$$

Nested Quantifiers

Type Terms

ι_i	are type constants
α, β	are type variables
σ, τ	are type terms
μ, ν	are closed type terms

ι_i	are terms	$FV(\iota_i) = \emptyset$
α	is a term	$FV(\alpha) = \{\alpha\}$
$\sigma \to \tau$	is a term if σ, τ are	$FV(\sigma \to \tau) = FV(\sigma) \cup FV(\tau)$
$(\!\![a_i : \sigma_i]\!\!)$	is a term if σ_i are	$FV((\!\![a_i : \sigma_i]\!\!)) = \bigcup_i FV(\sigma_i)$
$[\![a_i : \sigma_i]\!]$	is a term if σ_i are	$FV([\![a_i : \sigma_i]\!]) = \bigcup_i FV(\sigma_i)$
$\forall \alpha . \sigma$	is a term if σ is	$FV(\forall \alpha . \sigma) = FV(\sigma) \setminus \{\alpha\}$
$\exists \alpha . \sigma$	is a term if σ is	$FV(\exists \alpha . \sigma) = FV(\sigma) \setminus \{\alpha\}$

σ is a <u>closed term</u> iff $FV(\sigma) = \emptyset$

$\sigma\{\alpha_1 \ldots \alpha_n\}$ is a term σ such that $FV(\sigma) \subseteq \{\alpha_1, \ldots, \alpha_n\}$

Type Assignment

$v : \mathcal{B}_i$ iff $v = \bot_V$ or $v \in \mathcal{B}_i$.

$v : \mu \to \nu$ iff $v = \bot_V$ or $v \in (V \to V)$ and $\forall u.\ u : \mu \Rightarrow (v \restriction V \to V)u : \nu$

$v : [\![a_i : \mu_i]\!]$ iff $v = \bot_V$ or $v \in (L \to V)$ and $\forall i.\ (v \restriction L \to V)a_i : \mu_i$.

$v : \langle\!\langle a_i : \mu_i \rangle\!\rangle$ iff $v = \bot_V$ or $v \in (L \times V)$ and $\exists i.\ (v \restriction L \times V) = \langle a_i, u \rangle$ with $u : \mu_i$.

$v : \forall a.\ \sigma\{a\}$ iff $\forall \mu.\ v : \sigma\{\mu/a\}$

$v : \exists a.\ \sigma\{a\}$ iff $\exists \mu.\ v : \sigma\{\mu/a\}$

Semantics of Closed Type Expressions

Def $\mathcal{D}[\![\mu]\!] = \{ v \in V \mid v : \mu \}$

$\mathcal{D}[\![\mathcal{B}_i]\!] = B_i \text{ in } V$

$\mathcal{D}[\![\mu \to \nu]\!] = \{ f \in V \to V \mid f(\mathcal{D}[\![\mu]\!]) \subseteq \mathcal{D}[\![\nu]\!] \} \text{ in } V$

$\mathcal{D}[\![(a_i : \mu_i)\mathcal{D}]\!] = \{ r \in L \to V \mid \forall i.\ r(a_i) \in \mathcal{D}[\![\mu_i]\!] \} \text{ in } V$

$\mathcal{D}[\![[\![a_i : \mu_i]\!]]\!] = \{ c \in L \times V \mid \exists i.\ c = \langle a_i, u \rangle \text{ and } u \in \mathcal{D}[\![\mu_i]\!] \} \text{ in } V$

$\mathcal{D}[\![\forall \alpha.\ \sigma\{\alpha\}]\!] = \bigcap_{\mu} \mathcal{D}[\![\sigma\{\mu/\alpha\}]\!]$

$\mathcal{D}[\![\exists \alpha.\ \sigma\{\alpha\}]\!] = \bigcup_{\mu} \mathcal{D}[\![\sigma\{\mu/\alpha\}]\!]$

Quantifier Laws

$$\mathcal{D}[\![\forall\alpha.\ \mu \to \sigma\{\alpha\}]\!] = \mathcal{D}[\![\mu \to \forall\alpha.\ \sigma\{\alpha\}]\!]$$

$$\mathcal{D}[\![\exists\alpha.\ \mu \to \sigma\{\alpha\}]\!] = \mathcal{D}[\![\mu \to \exists\alpha.\ \sigma\{\alpha\}]\!]$$

$$\mathcal{D}[\![\forall\alpha.\ \sigma\{\alpha\} \to \mu]\!] = \mathcal{D}[\![(\exists\alpha.\ \sigma\{\alpha\}) \to \mu]\!]$$

$$\mathcal{D}[\![\exists\alpha.\ \sigma\{\alpha\} \to \mu]\!] \subseteq \mathcal{D}[\![(\forall\alpha.\ \sigma\{\alpha\}) \to \mu]\!]$$

$$\mathcal{D}[\![\forall\alpha.\ (a:\sigma\{\alpha\};\ a_i:\mu_i)]\!] = \mathcal{D}[\![(a:\forall\alpha.\sigma\{\alpha\};\ a_i:\mu_i)]\!]$$

$$\mathcal{D}[\![\exists\alpha.\ (a:\sigma\{\alpha\};\ a_i:\mu_i)]\!] = \mathcal{D}[\![(a:\exists\alpha.\sigma\{\alpha\};\ a_i:\mu_i)]\!]$$

$$\mathcal{D}[\![\forall\alpha.\ [\![a:\sigma\{\alpha\};\ a_i:\mu_i]\!]]\!] = \mathcal{D}[\![[\![a:\forall\alpha.\sigma\{\alpha\};\ a_i:\mu_i]\!]]\!]$$

$$\mathcal{D}[\![\exists\alpha.\ [\![a:\sigma\{\alpha\};\ a_i:\mu_i]\!]]\!] = \mathcal{D}[\![[\![a:\exists\alpha.\sigma\{\alpha\};\ a_i:\mu_i]\!]]\!]$$

Conclusions

- Inheritance can be treated in a denotational semantics framework.
- Specialized forms of inheritance admit static typechecking.
- Polymorphism has not been treated (future work), but inheritance and polymorphism are semantically compatible (both being interpreted as domain intersections).