A complete realisability semantics for intersection types and arbitrary expansion variables

Fairouz Kamareddine, Karim Nour, Vincent Rahli and J. B. Wells

ULTRA group, MACS, Heriot Watt University, Edinburgh, UK

Université de Savoie, Campus Scientifique, 73378 Le Bourget du Lac, France

September 3, 2008

Background

- **Principal typing** allows type inference.
- **Intersection types** allow expressing polymorphism in a finite way.
- **Expansion** allows establishing the principal typing property in intersection type systems.
- **Expansion variables** (E-variables) simplify and help mechanise expansion.
- Expansion has been generalised to deal with other type constructors such as the bang (!) from linear logic.

Interest

- We would like to cast some light on expansion.
- We use **realisability semantics** (Kleene). Types are interpreted by sets of realisers: terms of variants of the λ-calculus.
- Our final aim is to provide a **complete** realisability semantics for an intersection type system with expansion.
- First we consider only the expansion variables without the expansion mechanism.
- The challenge is to devise the space of meanings for E-variables.

Syntax

In essence our syntax is as follows:

Terms: $M ::= x \mid (M\,N) \mid (\lambda x.M)$

Types: $T ::= a \mid \omega \mid T_1 \to T_2 \mid T_1 \sqcap T_2 \mid eT$

Type environments: $\Gamma = (x_1 : T_1, \ldots, x_n : T_n)$.

Typings: $\Phi ::= \langle \Gamma \vdash T \rangle$ Judgement: $M : \Phi$

$(x + y : \langle + : \mathrm{int} \to \mathrm{int} \to \mathrm{int} \sqcap \mathrm{real} \to \mathrm{real} \to \mathrm{real}, x : \mathrm{int}, y : \mathrm{int} \vdash \mathrm{int}\rangle))$

$(x + y : \langle + : \mathrm{int} \to \mathrm{int} \to \mathrm{int} \sqcap \mathrm{real} \to \mathrm{real} \to \mathrm{real}, x : \mathrm{real}, y : \mathrm{real} \vdash \mathrm{real}\rangle))$

Typing rules:

$$\frac{}{x : \langle (x : T) \vdash T \rangle}\ \mathrm{var} \qquad \frac{}{M : \langle () \vdash \omega \rangle}\ \omega$$

$$\frac{M : \langle \Gamma, (x : T_1) \vdash T_2 \rangle}{\lambda x.M : \langle \Gamma \vdash T_1 \to T_2 \rangle}\ \mathrm{abs} \qquad \frac{M : \langle \Gamma_1 \vdash T_1 \rangle \quad M : \langle \Gamma_2 \vdash T_2 \rangle}{M : \langle \Gamma_1 \sqcap \Gamma_2 \vdash T_1 \sqcap T_2 \rangle}\ \sqcap$$

$$\frac{M_1 : \langle \Gamma_1 \vdash T_1 \to T_2 \rangle \quad M_2 : \langle \Gamma_2 \vdash T_1 \rangle}{M_1\ M_2 : \langle \Gamma_1 \sqcap \Gamma_2 \vdash T_2 \rangle}\ \mathrm{app} \qquad \frac{M : \langle \Gamma \vdash T \rangle}{M : \langle e\Gamma \vdash eT \rangle}\ \text{e-app}$$

Definition of some concepts - Principal Typing

- **Principal typing property:** A type system satisfies the principal typing property if for each typable term, there is a typing from which all other typings can be obtained via some set of operations

- **For example:** The simply typed lambda calculus, the simply type lambda sigma calculus, the simply typed lambda se calculus, have the Principal typing property.

Definition of some concepts - Principal Typing

- **Principal typing property:** A type system satisfies the principal typing property if for each typable term, there is a typing from which all other typings can be obtained via some set of operations
- **For example:** The simply typed lambda calculus, the simply type lambda sigma calculus, the simply typed lambda se calculus, have the Principal typing property.
- **For example:** $\lambda x.(y\ x) : \langle ?, ? \rangle$

Definition of some concepts - Principal Typing

- **Principal typing property:** A type system satisfies the principal typing property if for each typable term, there is a typing from which all other typings can be obtained via some set of operations
- **For example:** The simply typed lambda calculus, the simply type lambda sigma calculus, the simply typed lambda se calculus, have the Principal typing property.
- **For example:** $\lambda x.(y\ x) : \langle ?, ? \rangle$
  Possible typings: $\langle y{:}\alpha\to\beta, \alpha\to\beta \rangle$;
  $\langle y{:}\alpha\to\alpha, \alpha\to\alpha \rangle$; and many more
  Principal typing: $\langle y{:}\alpha\to\beta, \alpha\to\beta \rangle$

Definition of some concepts - Principal Typing

- **Principal typing property:** A type system satisfies the principal typing property if for each typable term, there is a typing from which all other typings can be obtained via some set of operations
- **For example:** The simply typed lambda calculus, the simply type lambda sigma calculus, the simply typed lambda se calculus, have the Principal typing property.
- **For example:** $\lambda x.(y\ x) : \langle ?, ? \rangle$
  Possible typings: $\langle y{:}\alpha \to \beta, \alpha \to \beta \rangle$;
  $\langle y{:}\alpha \to \alpha, \alpha \to \alpha \rangle$; and many more
  Principal typing: $\langle y{:}\alpha \to \beta, \alpha \to \beta \rangle$
- **For example:** $\lambda x.(y\ (y\ x)) : \langle ?, ? \rangle$

Definition of some concepts - Principal Typing

- **Principal typing property:** A type system satisfies the principal typing property if for each typable term, there is a typing from which all other typings can be obtained via some set of operations
- **For example:** The simply typed lambda calculus, the simply type lambda sigma calculus, the simply typed lambda se calculus, have the Principal typing property.
- **For example:** $\lambda x.(y\ x) : \langle ?, ? \rangle$
  Possible typings: $\langle y{:}\alpha{\to}\beta, \alpha{\to}\beta \rangle$;
  $\langle y{:}\alpha{\to}\alpha, \alpha{\to}\alpha \rangle$; and many more
  Principal typing: $\langle y{:}\alpha{\to}\beta, \alpha{\to}\beta \rangle$
- **For example:** $\lambda x.(y\ (y\ x)) : \langle ?, ? \rangle$
  Possible typings: $\langle y{:}\alpha{\to}\alpha, \alpha{\to}\alpha \rangle$;
  $\langle y{:}(\alpha{\to}\beta){\to}\alpha{\to}\beta, (\alpha{\to}\beta){\to}\alpha{\to}\beta \rangle$; and many more
  Principal typing: $\langle y{:}\alpha{\to}\alpha, \alpha{\to}\alpha \rangle$

Definition of some concepts - Intersection Types

- **Intersection type system:** Just like the $\forall$ quantifier, intersection types allow expressing polymorphism but in a finite way.

Definition of some concepts - Intersection Types

- **Intersection type system:** Just like the $\forall$ quantifier, intersection types allow expressing polymorphism but in a finite way.
- Intersection types are lists of usages: $\mathrm{int} \sqcap \mathrm{real}$

Definition of some concepts - Intersection Types

- **Intersection type system:** Just like the $\forall$ quantifier, intersection types allow expressing polymorphism but in a finite way.
- Intersection types are lists of usages: $\mathrm{int} \sqcap \mathrm{real}$
- In system F, the term $M = \lambda f.\lambda x.fx$ can be assigned the typing:

$$\Phi = \langle () \vdash \forall a.\forall b.(a \rightarrow b) \rightarrow a \rightarrow b \rangle$$

Definition of some concepts - Intersection Types

- **Intersection type system:** Just like the $\forall$ quantifier, intersection types allow expressing polymorphism but in a finite way.
- Intersection types are lists of usages: $\mathrm{int} \sqcap \mathrm{real}$
- In system F, the term $M = \lambda f.\lambda x.fx$ can be assigned the typing:

$$\Phi = \langle () \vdash \forall a.\forall b.(a \to b) \to a \to b \rangle$$

- But one might only need to use this term when its first argument f is a function which returns a string from an int or a real.

Definition of some concepts - Intersection Types

- **Intersection type system:** Just like the $\forall$ quantifier, intersection types allow expressing polymorphism but in a finite way.
- Intersection types are lists of usages: $\mathrm{int} \sqcap \mathrm{real}$
- In system F, the term $M = \lambda f.\lambda x.fx$ can be assigned the typing:

$$\Phi = \langle () \vdash \forall a.\forall b.(a \to b) \to a \to b \rangle$$

- But one might only need to use this term when its first argument f is a function which returns a string from an int or a real.
- In an intersection type system M can be assigned the typing:

$$\Phi' = \langle () \vdash (\mathrm{int} \sqcap \mathrm{real} \to \mathrm{string}) \to \mathrm{int} \sqcap \mathrm{real} \to \mathrm{string} \rangle$$

Definition of some concepts - Expansion

- Due to the ramification of the intersection types, the usual operations (substitution, weakening) are not enough anymore to obtain any typing of a typable (in an intersection type system) term from a principal one.

- **Expansion:** Introduced by Coppo, Dezani and Venneri [CDCV80] in order to restore the principal typing property in such systems (extensively improved by Carlier and Wells (2008) [Car08])

Expansion - example

The λ-term: $M = (\lambda x.x(\lambda y.yz))$

can be assigned the two following typings:

$$\Phi_1 = \langle (z : a) \vdash (((a \to b) \to b) \to c) \to c \rangle \qquad \text{(principal)}$$

$$\Phi_2 = \langle (z : a_1 \sqcap a_2) \vdash (((a_1 \to b_1) \to b_1) \sqcap ((a_2 \to b_2) \to b_2) \to c) \to c \rangle$$

An expansion operation can obtain Φ_2 from Φ_1

In System E (Carlier et al. [CPWK04]), the typing Φ_1 is replaced by:

$$\Phi_3 = \langle (z : ea) \vdash (e((a \to b) \to b) \to c) \to c \rangle$$

Φ_2 can be obtained from Φ_3 by substituting for e the expansion term:

$$E = (a := a_1, b := b_1) \sqcap (a := a_2, b := b_2)$$

Our aim

Our aim is to find a complete realisability semantics for an intersection type system with expansion variables.

How do we do that?

- Design of a calculus aiming at the capture of the meaning of an expansion variable: the encapsulation of a type.
  - λ-calculus indexed with natural numbers/list of natural numbers
- Design of a suitable type interpretation.
  - An expansion variable makes the realisers change level.
- Proof of the soundness and completeness of the semantics w.r.t. a given type system.

Previous work [KNRW08]: a first attempt (1)

- In order for our semantics to be informative enough, we have to distinguish the interpretations of T and eT where T is a type and e is an expansion variable.
- We used a labelled calculus were variables are labelled by integers (their levels).
- The application of an expansion variable to the typing of a term increases the level of the term:

$$\frac{M : \langle \Gamma \vdash T \rangle}{M^+ : \langle e\Gamma \vdash eT \rangle}\ (exp) \qquad \left(example : \frac{y^0 x^0 : \langle (y^0 : a \to b, x^0 : a) \vdash b \rangle}{y^1 x^1 : \langle (y^1 : e(a \to b), x^1 : ea) \vdash eb \rangle} \right)$$

- The interpretation of an expansion variable applied to a type allows to increase the level of the realisers of the type:

$$\mathcal{I}(eT) = \mathcal{I}(T)^+$$

- Hence, $M \in \mathcal{I}(T)$ implies $M^+ \in \mathcal{I}(T)^+ = \mathcal{I}(eT)$

Previous work [KNRW08]: a first attempt (2)

- However, the semantics of [KNRW08] did not distinguish between different expansion variables:

$$\mathcal{I}(e_1 T) = \mathcal{I}(e_2 T)$$

- Our semantics was not complete when considering more than one expansion variable:

$$M = \lambda x^0 . x^0$$

  is in the interpretation of

$$T = (e_1 a \rightarrow a) \rightarrow (e_2 a \rightarrow a)$$

  but M is not typable by T.

- In [KNRW08], we showed that our semantics was only complete if a single expansion variable is used.

Previous work [KNRW08]: a first attempt (3)

- Even more, since any term M has type ω, the above semantics of [KNRW08] cannot handle ω (since ω would need to belong to every possible level).
- So, [KNRW08] dropped ω completely.
- The absence of ω meant that we cannot deal with the whole λ-calculus.
- We restricted the calculus in [KNRW08] to a labelled version of the λI-calculus.

Our solution in this paper (1)

- We have seen in [KNRW08] that if the system has more than one expansion variable then it is difficult to give it a complete realisability semantics.
- We have also seen in [KNRW08] that a semantics based on single indices cannot deal with the universal type ω.
- In order to distinguish between different expansion variables, we need to modify the realisers.
- The interpretation of an expansion variable applied to a type should not only allow to increase the level of the realisers of the type, but should also depend on the considered expansion variable.

Our solution in this paper (2)

- To each expansion variable we associate a unique label: an integer. For simplicity: we associate i to e_i.
- The interpretation of an expansion variable applied to a type allows to increase the level of the realisers of the type w.r.t. the label associated to the expansion variable.
- We consider a set of labelled terms $\mathcal{M}$ where labels are lists of integers (each of them corresponding to an expansion variable). Example:

$$\lambda x^{(1,3,2)}.\lambda y^{(1)}.z^{\oslash} y^{(1)} x^{(1,3,2)}$$

- The set of terms of the untyped λ-calculus lives at each level L: $\mathcal{M}^L$.

Our solution in this paper (3)

The calculus is a labelled λ-calculus using lists of integers.

We impose some restrictions on terms such as:

- We do not allow a variable to coexist at different levels in a term: if $x^L, x^K \in \mathrm{fv}(M)$ then $L = K$. This is ensured by the $\diamond$ property.
  - We impose this restriction in order to have a suitable relation with a non-labelled typed λ-calculus.
  - Without this restriction we would face problems. For example (assume a function which erases the labels) :
    $\overline{(\lambda x^{(2)}.x^{\oslash} x^{(2)}) y^{(2)}} = (\lambda x.xx)y \rightarrow_\beta yy$ but
    $(\lambda x^{(2)}.x^{\oslash} x^{(2)}) y^{(2)} \rightarrow_\beta x^{\oslash} y^{(2)}$ and $\overline{x^{\oslash} y^{(2)}} = xy$.
- In MN we impose the level of M to be greater than the level of N.
  - This restriction is related to the typing of the terms.

We modify slightly the β-reduction rule to adapt it to our calculus.

Our solution in this paper (4)

$$e \in \mathcal{E} = \{\overline{e}_0, \overline{e}_1, \dots\} \qquad a \in \mathcal{A}$$
$$T \in \mathbb{T} ::= a \mid U \to T \qquad U \in \mathbb{U} ::= T \mid U_1 \sqcap U_2 \mid \omega^L \mid eU$$

$$\frac{}{x^{\oslash} : \langle (x^{\oslash} : T) \vdash T \rangle}\ (ax)$$

$$\frac{}{M : \langle env^{\omega}_{M} \vdash \omega^{\deg(M)} \rangle}\ (\omega)$$

$$\frac{M : \langle \Gamma, (x^L : U) \vdash T \rangle}{\lambda x^L.M : \langle \Gamma \vdash U \to T \rangle}\ (\to_I)$$

$$\frac{M : \langle \Gamma \vdash T \rangle \quad x^L \notin \mathrm{dom}(\Gamma)}{\lambda x^L.M : \langle \Gamma \vdash \omega^L \to T \rangle}\ (\to_I')$$

$$\frac{M_1 : \langle \Gamma_1 \vdash U \to T \rangle \quad M_2 : \langle \Gamma_2 \vdash U \rangle \quad \Gamma_1 \diamond \Gamma_2}{M_1 M_2 : \langle \Gamma_1 \sqcap \Gamma_2 \vdash T \rangle}\ (\to_E)$$

$$\frac{M : \langle \Gamma \vdash U_1 \rangle \qquad M : \langle \Gamma \vdash U_2 \rangle}{M : \langle \Gamma \vdash U_1 \sqcap U_2 \rangle}\ (\sqcap_I)$$

$$\frac{M : \langle \Gamma \vdash U \rangle}{M^{+j} : \langle \overline{e}_j \Gamma \vdash \overline{e}_j U \rangle}\ (e)$$

$$\frac{M : \langle \Gamma \vdash U \rangle \qquad \langle \Gamma \vdash U \rangle \sqsubseteq \langle \Gamma' \vdash U' \rangle}{M : \langle \Gamma' \vdash U' \rangle}\ (\sqsubseteq)$$

$$\frac{}{\Phi \sqsubseteq \Phi}\ (ref)$$

$$\frac{\Phi_1 \sqsubseteq \Phi_2 \quad \Phi_2 \sqsubseteq \Phi_3}{\Phi_1 \sqsubseteq \Phi_3}\ (tr)$$

$$\frac{d(U_1) = d(U_2)}{U_1 \sqcap U_2 \sqsubseteq U_1}\ (\sqcap_E)$$

$$\frac{U_1 \sqsubseteq V_1 \quad U_2 \sqsubseteq V_2}{U_1 \sqcap U_2 \sqsubseteq V_1 \sqcap V_2}\ (\sqcap)$$

$$\frac{U_2 \sqsubseteq U_1 \quad T_1 \sqsubseteq T_2}{U_1 \to T_1 \sqsubseteq U_2 \to T_2}\ (\to)$$

$$\frac{U_1 \sqsubseteq U_2}{eU_1 \sqsubseteq eU_2}\ (\sqsubseteq_e)$$

$$\frac{U_1 \sqsubseteq U_2}{\Gamma, y^L : U_1 \sqsubseteq \Gamma, y^L : U_2}\ (\sqsubseteq_c)$$

$$\frac{U_1 \sqsubseteq U_2 \quad \Gamma_2 \sqsubseteq \Gamma_1}{\langle \Gamma_1 \vdash U_1 \rangle \sqsubseteq \langle \Gamma_2 \vdash U_2 \rangle}\ (\sqsubseteq_{\langle\rangle})$$

Our solution in this paper (5)

Example of a type derivation in our type system:

- $v^{\oslash} v^{\oslash} : \langle v^{\oslash} : a \sqcap (a \to b) \vdash b \rangle$
- $v^{(0)} v^{(0)} : \langle v^{(0)} : \overline{e}_0 (a \sqcap (a \to b)) \vdash \overline{e}_0 b \rangle$
- $u^{\oslash} : \langle u^{\oslash} : \overline{e}_0 b \to c \vdash \overline{e}_0 b \to c \rangle$
- $u^{\oslash}(v^{(0)} v^{(0)}) : \langle u^{\oslash} : \overline{e}_0 b \to c, v^{(0)} : \overline{e}_0 (a \sqcap (a \to b)) \vdash c \rangle$
- $\lambda v^{(0)}.u^{\oslash}(v^{(0)} v^{(0)}) : \langle u^{\oslash} : \overline{e}_0 b \to c \vdash \overline{e}_0 (a \sqcap (a \to b)) \to c \rangle$
- $\lambda u^{\oslash}.\lambda v^{(0)}.u^{\oslash}(v^{(0)} v^{(0)}) : \langle () \vdash (\overline{e}_0 b \to c) \to (\overline{e}_0 (a \sqcap (a \to b)) \to c) \rangle$
- $\lambda u^{(1)}.\lambda v^{(1,0)}.u^{(1)}(v^{(1,0)} v^{(1,0)}) :$
  $\langle () \vdash \overline{e}_1 ((\overline{e}_0 b \to c) \to (\overline{e}_0 (a \sqcap (a \to b)) \to c)) \rangle$

Our solution in this paper (6)

Our semantics:

Let $r \in \{\beta, \beta\eta, h\}$.

- An r-interpretation $\mathcal{I} : \mathcal{A} \mapsto \mathcal{P}(\mathcal{M}^{\oslash})$ is a function such that for all $a \in \mathcal{A}$, $\mathcal{I}(a)$ satisfies simple properties such as saturation

  We extend an r-interpretation $\mathcal{I}$ to $\mathbb{U}$ as follows:
  - $\mathcal{I}(\omega^L) = \mathcal{M}^L$
  - $\mathcal{I}(\overline{e}_i U) = \mathcal{I}(U)^{+i}$
  - $\mathcal{I}(U_1 \sqcap U_2) = \mathcal{I}(U_1) \cap \mathcal{I}(U_2)$
  - $\mathcal{I}(U \to T) = \{M \in \mathcal{M} \mid \forall N \in \mathcal{I}(U).\ M \diamond N \Rightarrow MN \in \mathcal{I}(T)\}$

  Let $r\text{-int} = \{\mathcal{I} \mid \mathcal{I} \text{ is an } r\!-\!interpretation\}$.

- The meaning $[U]$ of the type U is defined as the intersection of the interpretations of U.

Our solution in this paper (7)

Our type system satisfies these properties (useful to prove the completeness of our semantics w.r.t. the type system):

- Subject reduction for β and η.
- Subject expansion for β.

We obtain soundness of our semantics w.r.t. the defined type system.

Soundness

Let $r \in \{\beta, \beta\eta, h\}$. If $M : \langle () \vdash U \rangle$, then $M \in [U]_r$

We obtain completeness of our semantics w.r.t. the defined type system.

Completeness

- $[U]_{\beta\eta} = \{M \in \mathcal{M}^L \; / \; M \text{ closed}, \; M \rhd^*_{\beta\eta} N \text{ and } N : \langle () \vdash U \rangle\}$.
- $[U]_\beta = [U]_h = \{M \in \mathcal{M}^L \; / \; M : \langle () \vdash U \rangle\}$.
- $[U]_{\beta\eta}$ is stable by reduction. I.e., If $M \in [U]_{\beta\eta}$ and $M \rhd^*_{\beta\eta} N$ then $N \in [U]_{\beta\eta}$.

Conclusion and Future work

- Expansion may be viewed to work like a multi-layered simultaneous substitution.
- Expansion is a crucial part of a procedure for calculating principal typings and helps support compositional type inference.
- In this paper we gave a complete semantics for intersection type systems with expansion variables.
- In order to overcome the problems of completeness, we changed our realisability semantics from one which uses indices as natural numbers to one that uses the indices as lists of natural numbers.
- As far as we know, our work constitutes the first study of a semantics of intersection type systems with E-variables (using realisability or any other approach) and of the difficulties involved.

Conclusion and Future work

In this paper we attempted to answer the questions:

- What does an expansion variable applied to a type stand for?
- What are the realisers of such a type?
- How can the relation between terms and types be described w.r.t. a type system?
- How can we extend realisability models to a type system with expansion?

Conclusion and Future work

- We now have to consider the expansion mechanism as well.
- The study of other semantics than realisability might provide further useful information on expansion.

Sébastien Carlier.

Expansion Algebra: a Foundational Theory with Applications to Type Systems and Type-Based Program Analysis.
PhD thesis, Heriot Watt University, School of Mathematical and Computing Sciences, 2008.

M. Coppo, M. Dezani-Ciancaglini, and B. Venneri.

Principal type schemes and λ-calculus semantic.
In J. R[oger] Hindley and J[onathan] P. Seldin, editors, *Sel+Hin:HBC-1980*, pages 535–560. Academic Press, 1980.

Sébastien Carlier, Jeff Polakow, J. B. Wells, and A. J. Kfoury.

System E: Expansion variables for flexible typing with linear and non-linear types and intersection types.
In *Programming Languages and Systems, 13th European Symposium on Programming, ESOP 2004*, pages 294–309, 2004.

Fairouz Kamareddine, Karim Nour, Vincent Rahli, and Joe B. Wells.

Developing realisability semantics for intersection types and expansion variables.
Presented to ITRS'08, 4th Workshop on Intersection Types and Related Systems, Turin, Italy, 25 March 2008, 2008.