SystemFI: A Core Language for Delegation-based Programming

December 10, 2014

Abstract

This paper ...

1 Introduction

Intersection types are useful for functional programming. In particular for extensibility and allowing new forms of composition.

We show a polymorphic language with intersection types and records, and how this language can be used to solve various common tasks in functional programming in a nicer way.

We have a formalization + implementation + proofs.

Prototype-based programming is one of the two major styles of object-oriented programming, the other being class-based programming which is featured in languages such as Java and C#. It has gained increasing popularity recently with the prominence of JavaScript in web applications. Prototype-based programming supports highly dynamic behaviors at run time that are not possible with traditional class-based programming. However, despite its flexibility, prototype-based programming is often criticized over concerns of correctness and safety. Furthermore, almost all prototype-based systems rely on the fact that the language is dynamically typed and interpreted.

This paper introduces System F_{IR}

Inheritance based

2 Overview

There should be a section informally describing the language (System FI) through various examples.

Mention properties informally via examples:

2.1 Algebras

2.2 Lenses

2.3 Embedded DSLs

3 System FI

Syntax + Type System

Talk about properties of the languages (commutativity of intersection types, ...)

4 Type-Directed Translation to System F

Type-Directed Translation to System F. Main results: type-preservation + coherence.

5 Implementation

Talk about the implementation + extensions not in the formalization.

6 Case Study?

7 Properties

Commutative $A \wedge B = B \wedge A$

$A = A \wedge A$
Idempotent
Source subtyping
Exclusion of 0-ary intersection =¿ dynamic typing
subtyping of the arrow
Algebra's & mixins lenses

8 Case Studies

9 Related work

Write a paragraph.

Dunfield: no records The idea of encoding records using intersection types is due to ... Reynolds and Castagna. The issue of coherence. Make a note about how part of this problem is mitigated.

Nystrom et. al. OOPSLA 06

Applications:

- Object/Fold Algebras. How to support extensibility in an easier way.

See Datatypes a la Carte

- Mixins

- Lenses? Can intersection types help with lenses? Perhaps making the types more natural and easy to understand/use?

- Embedded DSLs? Extensibility in DSLs? Composing multiple DSL interpretations?

http://www.cs.ox.ac.uk/jeremy.gibbons/publications/embedding.pdf

$\boxed{|\tau| = T}$

$$
\begin{array}{rcl}
|\alpha| & = & \alpha \\
|\tau_1 \to \tau_2| & = & |\tau_1| \to |\tau_2| \\
|\forall \alpha . \tau| & = & \forall \alpha . |\tau| \\
|t_1 \& t_2| & = & \langle |\tau_1|, |\tau_2| \rangle \\
|\{l : \tau\}| & = & |\tau|
\end{array}
$$

Lemma 1. *If*

$$\Gamma \vdash \tau_1 <: \tau_2 \hookrightarrow C$$

then

$$|\Gamma| \vdash C : |\tau_1| \to |\tau_2|$$

Proof. By structural induction on the types and the corresponding inference rule.

`(SubVar)`
`(SubFun)`
`(SubForall)`
`(SubAnd1)`
`(SubAnd2)`
`(SubAnd3)`
`(SubRcd)`

□

Lemma 2. *If*

$$\Gamma \vdash_{get} \tau; l = C; \tau_1$$

then

$$|\Gamma| \vdash C : |\tau| \to |\tau_1|$$

Proof. By structural induction on the type and the corresponding inference rule.

`(Get-Base)` $\Gamma \vdash_{get} \{l : \tau\}; l = \lambda(x : |\{l : \tau\}|).x; \tau$

By the induction hypothesis

$$|\Gamma| \vdash \lambda(x : |\{l : \tau\}|).x : |\{l : \tau\}| \to |\tau|$$

`(Get-Left)`
`(Get-Right)`

□

Lemma 3. *If*

$$\Gamma \vdash_{put} \tau; l; E = C; \tau_1$$

then

$$|\Gamma| \vdash C : |\tau| \to |\tau|$$

Proof. By structural induction on the type and the corresponding inference rule.

`(Put-Base)`
`(Put-Left)`

(Put-Right)

∎

Lemma 4. *If*

$$\Gamma \vdash \tau$$

then

$$|\Gamma| \vdash |\tau|$$

Proof. Since

$$\Gamma \vdash \tau$$

It follows from (FI-WF) that

$$\mathrm{ftv}(\tau) \subseteq \mathrm{ftv}(\Gamma)$$

And hence

$$\mathrm{ftv}(|\tau|) \subseteq \mathrm{ftv}(|\Gamma|)$$

By (F-WF) we have

$$\Gamma \vdash \tau$$

∎

Theorem 1 (Type preserving translation)**.** *If*

$$\Gamma \vdash e : \tau \hookrightarrow E$$

then

$$|\Gamma| \vdash E : |\tau|$$

Proof. By structural induction on the expression and the corresponding inference rule.

(TrVar) $\Gamma \vdash x : \tau \hookrightarrow x$

It follows from (TrVar) that

$$(x : \tau) \in \Gamma$$

Based on the definition of $|\cdot|$,

$$(x : |\tau|) \in |\Gamma|$$

Thus we have by (F-Var) that

$$|\Gamma| \vdash x : |\tau|$$

(TrAbs) $\Gamma \vdash \lambda(x : \tau_1).e : \tau_1 \to \tau_2 \hookrightarrow \lambda x : |\tau_1|.E$

It follows from (TrAbs) that

$$\Gamma, x : \tau_1 \vdash e : \tau_2 \hookrightarrow E$$

And by the induction hypothesis that

$$|\Gamma|, x : |\tau_1| \vdash E : |\tau_2|$$

By (TrAbs) we also have

$$\Gamma \vdash \tau_1$$

It follows from Lemma **??** that

$$|\Gamma| \vdash |\tau_1|$$

Hence by (F-Abs) and the definition of $|\cdot|$ we have

$$|\Gamma| \vdash \lambda x : |\tau_1|.E : |\tau_1 \to \tau_2|$$

(TrApp) $\Gamma \vdash e_1 e_2 : \tau_2 \hookrightarrow E_1(CE_2)$

From (TrApp) we have

$$\Gamma \vdash \tau_3 <: \tau_1 \hookrightarrow C$$

Applying Lemma **??** to the above we have

$$|\Gamma| \vdash C : |\tau_3| \to |\tau_1|$$

Also from (TrApp) and the induction hypothesis

$$|\Gamma| \vdash E_1 : |\tau_1| \to |\tau_2|$$

Also from (TrApp) and the induction hypothesis

$$|\Gamma| \vdash E_2 : |\tau_3|$$

Assembling those parts using (F-App) we come to

$$|\Gamma| \vdash E_1(CE_2) : |\tau_2|$$

∎

(TrTAbs) $\Gamma \vdash \Lambda\alpha.e : \forall\alpha.\tau \hookrightarrow \forall\alpha.E$

From (TrTAbs) we have

$$\Gamma \vdash e : \tau \hookrightarrow E$$

By the induction hypothesis we have

$$|\Gamma| \vdash E : |\tau|$$

Thus by `(F-TAbs)` and the definition of $|\cdot|$

$$\Gamma \vdash \Lambda\alpha.E : |\forall\alpha.\tau|$$

`(TrTApp)` $\Gamma \vdash e\ \tau : [\alpha := \tau]\tau_1 \hookrightarrow E\ |\tau|$

From `(TrTApp)` we have

$$\Gamma \vdash e : \forall\alpha.\tau_1 \hookrightarrow E$$

And by the induction hypothesis that

$$|\Gamma| \vdash E : \forall\alpha.|\tau_1|$$

Also from `(TrTApp)` and Lemma **??** we have

$$|\Gamma| \vdash |\tau|$$

Then by `(F-TApp)` that

$$|\Gamma| \vdash E\ |\tau| : [\alpha := |\tau|]|\tau_1|$$

Therefore

$$|\Gamma| \vdash E\ |\tau| : |[\alpha := \tau]|\tau_1||$$

`(TrMerge)` $\Gamma \vdash e_1,, e_2 : \tau_1 \& \tau_2 \hookrightarrow \langle E1, E2\rangle$

From `(TrMerge)` and the induction hypothesis we have

$$|\Gamma| \vdash E_1 : |\tau_1|$$

and

$$|\Gamma| \vdash E_2 : |\tau_2|$$

Hence by `(F-Pair)`

$$|\Gamma| \vdash \langle E_1, E_2\rangle : \langle|\tau_1|, |\tau_2|\rangle$$

Hence by the definition of $|\cdot|$

$$|\Gamma| \vdash \langle E_1, E_2\rangle : |\tau_1 \& \tau_2|$$

`(TrRcdIntro)` $\Gamma \vdash \{l = e\} : \{l : \tau\} \hookrightarrow E$

From `(TrRcdIntro)` we have

$$\Gamma \vdash e : \tau \hookrightarrow E$$

And by the induction hypothesis that

$$|\Gamma| \vdash E : |\tau|$$

Thus by the definition of $|\cdot|$

$$|\Gamma| \vdash E : |\{l : \tau\}|$$

`(TrRcdElim)` $\Gamma \vdash e.l : \tau_1 \hookrightarrow CE$

From `(TrRcdElim)`

$$\Gamma \vdash e : \tau \hookrightarrow E$$

And by the induction hypothesis that

$$|\Gamma| \vdash E : |\tau|$$

Also from `(TrRcdEim)`

$$\Gamma \vdash_{get} e; l = C; \tau_1$$

Applying Lemma **??** to the above we have

$$|\Gamma| \vdash C : |\tau| \to |\tau_1|$$

Hence by `(F-App)` we have

$$|\Gamma| \vdash CE : |\tau_1|$$

`(TrRcdUpd)` $\Gamma \vdash e \text{ with } \{l = e_1\} : \tau \hookrightarrow CE$

From `(TrRcdUpd)`

$$\Gamma \vdash e : \tau \hookrightarrow E$$

And by the induction hypothesis that

$$|\Gamma| \vdash E : |\tau|$$

Also from `(TrRcdUpd)`

$$\Gamma \vdash_{put} t; l; E = C; \tau_1$$

Applying Lemma **??** to the above we have

$$|\Gamma| \vdash C : |\tau| \to |\tau|$$

Hence by `(F-App)` we have

$$|\Gamma| \vdash CE : |\tau|$$