Consistency of static and dynamic semantics

Jin Cao

September 12, 2008
Overview

- In this project, we use the theory of Scott-domains to provide a denotational model of co-inductive data types.
- The overall goal is to use this model to give a proof of the consistency of the static and the dynamic relational semantics of a small functional programming language with recursive definitions.
- The project is based on the article ”Co-induction in relational semantics” by Robin Milner and Mads Tofte.
- The novelty in our project is that, instead of using Peter Aczel’s non-well founded set theory, we work in the simpler theory of Scott-domains within ordinary set theory as meta-theory.
Dynamic and static semantics of the functional programming language Exp

- Exp is a small functional language whose expressions are built from variables, constants, \( \lambda \)-abstraction, application and recursion.
- The dynamic semantics is given by a big-step relation

\[ E \vdash exp \rightarrow v \]

relating expressions to values.
- The static semantics is given by typing rules of the form

\[ TE \vdash exp \Rightarrow \tau \]

relating an expression \( exp \) with a simple type in a typing context \( TE \).
The dynamic semantics uses the concept of a closure, as the values of \( \lambda \)-abstraction and recursive functions.

In particular, the value assigned to a recursive function

$$fix\; f(x) = \text{exp}$$

must be a closure \( cl_{\infty} \) satisfying a recursive equation

$$cl_{\infty} = \langle x, \text{exp}, E + \{f \mapsto cl_{\infty}\} \rangle.$$
The solution of recursive equation

The recursive equation \( cl_\infty = \langle x, \exp, E + \{ f \mapsto cl_\infty \} \rangle \) require an infinite descending \( \omega \)-chain \( cl_\infty \ni \ldots \ni cl_\infty \in \ldots \), which contradicts the Axiom of Foundation in well founded set theory.

Milner and Tofte solve this equation by using non-well founded sets as introduced by Peter Aczel, where the Axiom of Foundation was dropped. Hence the contradiction omitted.

In this project we will use \textit{Scott-domains} instead of non-well founded set.
Data in Set theory versus Domain theory

Set: \[ \langle A, B \rangle := \{ \{A\}, \{A, B\} \}. \]
Therefore, \[ \langle A, B \rangle \ni \{A, B\} \ni B. \]

Domains: \[ x \in D, \ y \in E. \]
\[ x \simeq \hat{x} := \{ x_0 \in D \mid x_0 \sqsubseteq x, x_0 \text{ compact} \}, \]
\[ y \simeq \hat{y} := \{ y_0 \in E \mid y_0 \sqsubseteq y, y_0 \text{ compact} \}. \]
Therefore, \[ (x, y) \simeq (\hat{x}, \hat{y}) = \{ \langle x_0, y_0 \rangle \mid x_0 \in \hat{x}, y_0 \in \hat{y} \} \]
The consistency problem

The statement of the consistency of static and dynamic semantics:

If \( E \vdash \text{exp} \rightarrow c \)
and \( TE \vdash \text{exp} \Rightarrow \tau \)
and \( E \text{ IsOf } TE \)
Then \( c \text{ IsOf } \tau \).
Syntax of Exp

\[
\text{exp ::= } \begin{cases} 
  x & \text{Variable,} \\
  c & \text{Constant,} \\
  \text{fn } x \Rightarrow \text{exp} & \text{Abstraction,} \\
  \text{fix } f(x) = \text{exp} & \text{Recursive function,} \\
  \text{exp}_1 \text{ exp}_2 & \text{Application.}
\end{cases}
\]
Dynamic semantics

For the dynamic semantics we need the sets $\text{Const}$, $\text{Var}$ and domains $\text{Val}$, $\text{Env}$, $\text{Clos}$, these domains satisfy the following equations.

\[
\nu \in \text{Val} = \text{Const}_\bot \oplus \text{Clos} \quad \text{Values}
\]

\[
E \in \text{Env} = \text{Var}_\bot \overset{\text{fin}}{\longrightarrow} \text{Val} \quad \text{Environments}
\]

\[
cl \text{ or } \langle x, \exp, E \rangle \in \text{Clos} = \text{Var}_\bot \times \text{Exp}_\bot \times \text{Env} \quad \text{Closures}
\]

Finally, we assume that we are given a partial function

\[
\text{APPLY} : \text{Const} \times \text{Const} \rightarrow \text{Const}
\]

defining the meaning of primitive operations and constants that can be part of the calculus.
Inference rules of the dynamic semantics

The dynamic semantics of Exp is a relational semantics given by a set of judgements of the form

\[ E \vdash exp \rightarrow v, \]

(“expression exp evaluates to value v in environment E”)

Judgements: ”big step” or ”natural” semantics.

\[ E \vdash c \rightarrow c \quad \text{(Constant)} \]

\[ \frac{x \in \text{Dom } E}{E \vdash x \rightarrow E(x)} \quad \text{(Variables)} \]

\[ E \vdash \text{fn } x \Rightarrow exp \rightarrow \langle x, \ exp, \ E \rangle \quad \text{(Abstraction)} \]

\[ cl_\infty = \langle x, \ exp, \ E + \{f \mapsto cl_\infty \} \rangle \quad \text{(Recursion)} \]

\[ E \vdash \text{fix } f(x) = exp \rightarrow cl_\infty \]
Inference rules of the dynamic semantics (continued)

\[
\begin{align*}
E \vdash \text{exp}_1 & \rightarrow c_1 & E \vdash \text{exp}_2 & \rightarrow c_2 & c = \text{APPLY}(c_1, c_2) \\
E \vdash \text{exp}_1 \text{exp}_2 & \rightarrow c
\end{align*}
\]

(Application of constant)

\[
\begin{align*}
E \vdash \text{exp}_1 & \rightarrow \langle x', \text{exp}', E' \rangle \\
E \vdash \text{exp}_2 & \rightarrow v_2 \\
E' + \{x' \mapsto v_2\} \vdash \text{exp}' & \rightarrow v \\
E \vdash \text{exp}_1 \text{exp}_2 & \rightarrow v
\end{align*}
\]

(Application of Closure)
Static semantics

The *static semantics* of $\text{Exp}$ is defined by a simple monomorphic type inference system.

Type expression:

$$\tau ::= \pi \mid \tau_1 \rightarrow \tau_2 \quad (\pi \text{ is a primitive type})$$

A type environment is a finite map from variables to types:

$$TE \in \text{TyEnv} = \text{Var}^{\text{fin}} \rightarrow \text{Type}$$

We assume we are given a relation

$$\text{IsOf} \subseteq \text{Const} \times \text{Type}$$

that relates a constant to a type, for example, 1 to int, such that
If $c = \text{APPLY}(c_1, c_2)$ and $c_1 \text{ IsOf } (\tau_1 \rightarrow \tau_2)$ and $c_2 \text{ IsOf } \tau_1$ then $c \text{ IsOf } \tau_2$. 
Inference rules of the static semantics

\[ TE \vdash \exp \rightarrowarrow \tau \]

("\exp\) elaborates to \(\tau\) in \(TE\)."

\[ \frac{c \text{ IsOf } \tau}{TE \vdash c \rightarrowarrow \tau} \quad \text{(Constant)} \]

\[ \frac{x \in \text{Dom} \; TE}{TE \vdash x \rightarrowarrow TE(x)} \quad \text{(Variables)} \]

\[ \frac{TE + \{x \vdash \tau_1\} \vdash \exp \rightarrowarrow \tau_2}{TE \vdash \text{fn } x \Rightarrow \exp \rightarrowarrow \tau_1 \rightarrow \tau_2} \quad \text{(Abstraction)} \]
Inference rules of the static semantics (continued)

\[
TE + \{ f \vdash \tau_1 \rightarrow \tau_2 \} + \{ x \vdash \tau_1 \} \vdash exp \Rightarrow \tau_2 \\
\]

\[
TE \vdash fix f(x) = exp \Rightarrow \tau_1 \rightarrow \tau_2
\]

(Recursion)

\[
TE \vdash exp_1 \Rightarrow \tau_1 \rightarrow \tau_2 \quad TE \vdash exp_2 \Rightarrow \tau_1 \\
\]

\[
TE \vdash exp_1 \ exp_2 \Rightarrow \tau_2
\]

(Application)
Using Domains to model the semantics

- The date types Val, Clos, Env e.t.c are modeled as domains. (as the solution of recursive domain equations)
- The closure $c_{l_{\infty}}$ is obtained as the least fixed point of a continuous function

$$F : \text{Clos} \rightarrow \text{Clos}.$$ 

$$c_{l_{\infty}} = \langle x, \text{exp}, E + \{ f \mapsto c_{l_{\infty}} \} \rangle = F(c_{l_{\infty}})$$
Proof of the consistency theorem

First, we informally define what it means for a value to have a type.

\[ v : \tau \text{ iff } \begin{cases} 
(i) & \text{if } v = c \text{ then } v \text{ IsOf } \tau; \\
(ii) & \text{if } v = \langle x, \exp, E \rangle, \text{ then there exists a } TE \text{ such that } \\
& \quad TE \vdash (\text{fn}x \Rightarrow \exp) \Rightarrow \tau \text{ and } \text{Dom}(E) = \text{Dom}(TE) \\
& \quad \text{and } E(x) : TE(x), \text{ for all } x \in \text{Dom}(E). 
\end{cases} \]

We view this as a coinductive definition of a relation

\[ : \subseteq \text{Val} \times \text{Type}. \]
Structure of the proof of the consistency theorem

First we prove the statement:

If $E : TE$ and $E \vdash exp \rightarrow v$ and $TE \vdash exp \Rightarrow \tau$, then $v : \tau$.

by induction on the definition of $E \vdash exp \rightarrow v$.

From this the consistency theorem:

If $E$ IsOf $TE$ and $E \vdash exp \rightarrow c$ and $TE \vdash exp \Rightarrow \tau$, then $c$ IsOf $\tau$.

follows because $c : \tau$ iff $c$ IsOf $\tau$. 
Least fixed point is not big enough

Next, we use as an example the factorial function to show that the least fixed point is not big enough to define the relation $\nu : \tau$. We show

$$ (cl_{fact}, \text{int} \rightarrow \text{int}) \in Q^{\text{max}} \setminus Q^{\text{min}}, $$

there are two parts in the proof.

1. $(cl_{fact}, \text{int} \rightarrow \text{int}) \in Q^{\text{max}}$, and
2. $(cl_{fact}, \text{int} \rightarrow \text{int}) \notin Q^{\text{min}}$.

Instead of defining $Q^{\text{min}}$ using ordinals as in the original paper, we prove the second goal by using the induction rule of Tarski’s fixed point theorem:

$$ \forall Q \subseteq U : F(\widetilde{Q}_{fact}) \subseteq \widetilde{Q}_{fact} \Rightarrow Q^{\text{min}} \subseteq \widetilde{Q}_{fact} $$

where $Q_{fact} = \{(cl_{fact}, \text{int} \rightarrow \text{int})\}$ and $\widetilde{Q}_{fact} = U \setminus Q_{fact}$. 
Conclusion

The main result of this project:

- We used domain theory instead of non-well founded sets to prove the consistency of the static and dynamic semantics of the programming language Exp.

The advantages of using domains:

- The foundation of mathematics (i.e. set theory) is not changed.
- Domains allow for the interpretation of a large variety of extensions of Exp. For example, one could introduce a recursive type $\tau_\infty$ with the equation

$$
\tau_\infty = \text{Int} + (\tau_\infty \rightarrow \tau_\infty)
$$

and the corresponding language constructs $\tau_\infty$ would be interpreted by the domain

$$
D \simeq \mathbb{N}_\perp + [D \rightarrow D].
$$

This would be difficult with non-well founded sets.