Notes from 3rd Meeting

June 30, 2016

1 Fixpoints

Fixpoints can be written as the following function

$$fix : (X \to X) \to X$$

which takes a function f to the element $x \in X$ where $f(x) = x$.

1.1 PCF

In PCF the transition rule for the fixpoint operation is:

$$fix\ x.e \mapsto [fix\ e/x]\ e$$

So for a function f applied to an argument x we have:

$$fix\ x.f(x) \mapsto f(fix\ x.e)$$

1.2 Fixpoint of Continuous Functions

Semanitcally, we can prove that there is a least fixpoint for any continuous function f and that this is the limit of the following chain:

$$\bot \sqsubseteq f(\bot) \sqsubseteq \ldots \sqsubseteq f^n(\bot)$$

Therefore our fixpoint is defined as:

$$fix(f) \equiv \sqcup^i f^i(\bot)$$

There are three things we need to prove:

1. **The limit of the chain exists**
2. **The limit is a fixpoint of f**
3. **This is the least fixpoint of f** To prove this we can use the fact that f is continuous, so for a chain

   $$\bot \sqsubseteq f(\bot) \sqsubseteq \ldots \sqsubseteq f^n(\bot)$$

   when you apply f you get

   $$f(\bot) \sqsubseteq f^2(\bot) \sqsubseteq \ldots \sqsubseteq f^{n+1}(\bot)$$

   Prove $f(\sqcup^i f^i(\bot)) = \sqcup^i f^{i+1}(\bot)$

2 Full Abstraction

Contextual Equivalence is defined as, for all contexts C, if an expression e evaluates to a value in a finite amount of steps, then so does e' and vice versa.

$$e \approx_{ctx} e' = (\forall C.\ C[e] \Downarrow \Leftrightarrow\ C[e'] \Downarrow)$$

where $\Downarrow$ means that either the expression terminates in a finite amount of steps or it loops forever, or formally:

$$e \Downarrow = \exists n, v.\ e \mapsto^n v$$

If the denotational semantics of two expressions are the same then they must be contextually equivalent and vice versa,

$$[\![e]\!] = [\![e']\!] \ \Leftrightarrow\ e \approx_{ctx} e'$$

We do not have contextual equivalence in PCF because we cannot define parallel-or in PCF in a way that we get the same semantics for every possible version of the function. However, we have $e \approx_{ctx} e$, because $\approx_{ctx}$ is reflexive. Therefore we cannot define Contextual Equivalence in PCF.

In our language from the mini project we could prove soundness and completeness because there were no types. They were the following theorems:

Soundness Anything definable in the operational semantics has a valid denotation. $e \mapsto^* n \Rightarrow [\![e]\!] = n$

Completeness Any expression with a valid denotation can be defined in the operational semantics. $[\![e]\!] = n \Rightarrow e \mapsto^* n$

These do not hold for higher types in PCF, so we do not have full abstraction and therefore cannot prove completeness.

3 PCF

The syntax of PCF is the following:

Types:
$A ::= Nat \mid A \to B$

Expressions
$e ::= \lambda x : A.e \mid e\ e' \mid x \mid z \mid s(e) \mid case\ (e, z \to e_0, s(x) \to e_S) \mid fix\ x : A.\ e$

Contexts
$\Gamma = \ldots \mid \Gamma, x : A$

Rule for contexts
$\Gamma \vdash e : A$

3.1 Operational Semantics of PCF

$e \mapsto e'$ (Call by Name)
$(\lambda x : A.\ e) \mapsto [e'/x]e$
$case\ (z, z \to e_0, s(x) \to e_S) \mapsto e_0$ $case\ (s(x), z \to e_0, s(x) \to e_S) \mapsto [e/x]e_s$ $fix\ x : A.e \mapsto [fix\ x : A.e/x]e$

3.2 Lemmas for Type Safety

Weakening If $\Gamma \vdash x : A$ then $\Gamma, x : C \vdash x : A$

Substitution If $\Gamma \vdash x : A$ and $\Gamma, x : A \vdash e : C$ then $\Gamma \vdash [e/x]e' : C$.

3.3 Type Safety Theorems

Type Preservation If $\Gamma \vdash x : A$ and $e \mapsto e'$ (in one step) then $\Gamma \vdash e' : A$

Type Progress If $\Gamma \vdash e : A$ then $e \mapsto e'$ or e is a value.

Possible values are
$v :: z \mid s(x) \mid \lambda x : A.\ e$
which are numbers or non-recursive functions.