SE547: Lecture 2

Overview

Foundational calculi

Lambda-calculus

Equivalence

Derived forms

Foundational calculi

What is a foundational calculus?

What are some examples of foundational calculi?

Why are they interesting?

Foundational calculi

Most foundational calculi come with:

1) Syntax of core language.

2) Dynamic semantics of core language.

3) Derived forms.

What are these?

Foundational calculi

Many problems in computing are *safety* properties.

What is a safety property?

What are example safety properties?

What are example properties which are not safety properties?

How can we give a formal definition of a safety properties?

Lambda-calculus

What are the goals of the lambda-calculus?

History: Schönfinkel (1920's), Church (1930's), McCarthy (1950's), Landin (1960's), Scheme / Standard ML / Haskell / CAML / ... (1970's-now).

Lambda-calculus

Assume a collection of variables x, y, z... Core syntax:

$$
\begin{aligned}
&L, M, N ::= \\
&\quad x \\
&\quad M\ N \\
&\quad \lambda x.M
\end{aligned}
$$

What are these?

Note: no booleans, integers, while loops, etc. Is this worrying?

Also note: no threads, concurrency controls, etc. Is this worrying?

Lambda-calculus

Examples:

1. $\lambda x \,.\, x$
2. $\lambda y \,.\, y$
3. $\lambda y \,.\, x$
4. $\lambda x \,.\, \lambda y \,.\, x$
5. $\lambda x \,.\, \lambda y \,.\, y$
6. $\lambda y \,.\, \lambda x \,.\, y$
7. $(\, \lambda x \,.\, x\,)(\, \lambda y \,.\, y\,)$
8. $(\, \lambda x \,.\, x\,(\, \lambda y \,.\, y\,)\,)$
9. $(\, \lambda x \,.\, x\,(\, \lambda y \,.\, y\,)\,)\,(\, \lambda z \,.\, z\,)$
10. $(\, \lambda x \,.\, x\; x\,)\,(\, \lambda x \,.\, x\; x\,)$

Which of these ‘are the same program’? What does that mean?

Equivalence

Two notions of ‘are the same program’:

1. alpha-equivalence: ‘allowed to rename bound variables’.
2. beta-equivalence: alpha + ‘allowed to apply functions’.

Which of the examples are alpha-equivalent? Which are beta-equivalent?

Equivalence

Formalize alpha-equivalence...

Define $[M/x]N$ as 'replace M for x in N'. Examples:

1. $[\ \lambda y\ .\ y\ /\ x\ ](\ x\ )$
2. $[\ \lambda z\ .\ z\ /\ x\ ]\ (\ x\ (\ \lambda y\ .\ y\ )\ )$
3. $[\ \lambda x\ .\ x\ x\ /\ x\ ]\ (\ x\ x\ )$

Alpha-equivalence is generated by:

$$(\ \lambda x\ .\ M\ ) = (\ \lambda y\ .\ [y/x]M\ ) \qquad \text{when } y \text{ is fresh}$$

Which of these are alpha-equivalent?

1. $\lambda x\ .\ \lambda y\ .\ x$
2. $\lambda x\ .\ \lambda y\ .\ y$
3. $\lambda y\ .\ \lambda x\ .\ y$

Equivalence

Formalize function application (jargon: beta-reduction) generated by:

$$(\lambda x \,.\, M)\, \mathrm{N} \rightarrow ([N/x]M)$$

Examples:

1. $(\lambda x \,.\, x)(\lambda y \,.\, y)$
2. $(\lambda x \,.\, x \,(\lambda y \,.\, y))$
3. $(\lambda x \,.\, x \,(\lambda y \,.\, y)) \,(\lambda z \,.\, z)$
4. $(\lambda x \,.\, x \, x) \,(\lambda x \,.\, x \, x)$

Equivalence

Beta-equivalence:

$$M =_\beta N \qquad \text{whenever} \qquad \exists\, L \,.\, M \rightarrow^* L \text{ and } N \rightarrow^* L$$

Sanity checks:

$M =_\beta M$?

If $M =_\beta N$ then $N =_\beta M$?

If $L =_\beta M$ and $M =_\beta N$ then $L =_\beta N$?

Derived forms

Booleans:

True = (λx . λy . x)
False = (λx . λy . y)
if L { M } else { N } = (L M N)

Verify:

if True { M } else { N } $=_\beta M$
if False { M } else { N } $=_\beta N$

Derived forms

Pairs:

$$(M, N) = (\lambda x\ .\ x\ M\ N)$$
$$\text{Fst} = \lambda z\ .\ \text{z}\ (\lambda x\ .\ \lambda y\ .\ x)$$
$$\text{Snd} = \lambda z\ .\ \text{z}\ (\lambda x\ .\ \lambda y\ .\ y)$$

Verify:

$$\text{Fst}\ (M, N) =_\beta M$$
$$\text{Snd}\ (M, N) =_\beta N$$

Similar codings for integers, lists, etc.

Derived forms

Recursion:

$$\text{fix } M = (\ \lambda x\ .\ M\ (\ x\ x\ )\ )\ (\ \lambda x\ .\ M\ (\ x\ x\ )\ )$$

Verify:

$$\text{fix } M =_\beta \text{M}\ (\ \text{fix } M\ )$$

Example:

$$\text{factorial} = \text{fix fact}$$
$$\text{fact} = (\ \lambda f\ .\ \lambda x\ .\ \text{if } (x < 2)\ \{\ 1\ \}\ \text{else}\ \{\ x * f\ (x - 1)\ \}\ )$$

Verify:

$$\text{factorial } 3 =_\beta 6$$

Next week

Homework sheet 2.

Calculi for protocols: pi- and spi-calculus.