Denotational Semantics

Renato Neves

Universidade do Minho

Semantics for Every Season

Operational semantics	How a program operates
Denotational semantics	What a program is
Axiomatic semantics	Which logical properties a program satisfies

Table of Contents

Motivation

My first denotational semantics

Denotational semantics for a while-language

Domain theory

Conclusions

Compiler Correctness and Contextual Equivalence

Adopted a natural notion of equivalence

$$\mathtt{p} \equiv_o \mathtt{q} \text{ iff } \Big(\text{for every } \sigma.\ \langle \mathtt{p}, \sigma \rangle \Downarrow \sigma' \text{ iff } \langle \mathtt{q}, \sigma \rangle \Downarrow \sigma'\Big)$$

Compiler Correctness and Contextual Equivalence

Adopted a natural notion of equivalence

$$\mathtt{p} \equiv_o \mathtt{q} \text{ iff } \Big(\text{for every } \sigma.\ \langle \mathtt{p}, \sigma\rangle \Downarrow \sigma' \text{ iff } \langle \mathtt{q}, \sigma\rangle \Downarrow \sigma'\Big)$$

Compilers adopt a stronger version

$$\mathtt{p} \equiv \mathtt{q} \text{ iff } \Big(\text{for every context } C.\ C[\mathtt{p}] \equiv_o C[\mathtt{q}]\Big)$$

Compiler Correctness and Contextual Equivalence

Adopted a natural notion of equivalence

$$\mathtt{p} \equiv_o \mathtt{q} \text{ iff } \Big(\text{for every } \sigma.\ \langle \mathtt{p}, \sigma\rangle \Downarrow \sigma' \text{ iff } \langle \mathtt{q}, \sigma\rangle \Downarrow \sigma'\Big)$$

Compilers adopt a stronger version

$$\mathtt{p} \equiv \mathtt{q} \text{ iff } \Big(\text{for every context } C.\ C[\mathtt{p}] \equiv_o C[\mathtt{q}]\Big)$$

Why ?

Contextual Equivalence

Contexts

$C ::= [-] \mid C \wedge \mathtt{b} \mid \mathtt{b} \wedge C \mid \neg C$

Exercise

Prove the equivalence $\mathtt{b}_1 \equiv_o \mathtt{b}_2$ iff $\mathtt{b}_1 \equiv \mathtt{b}_2$

Contextual Equivalence

Contexts

$C ::= [-] \mid C \wedge \mathtt{b} \mid \mathtt{b} \wedge C \mid \neg C$

Exercise

Prove the equivalence $\mathtt{b}_1 \equiv_o \mathtt{b}_2$ iff $\mathtt{b}_1 \equiv \mathtt{b}_2$

Exercise

Repeat the previous exercise now for arithmetic expressions

Contextual Equivalence

Contexts

$C ::= [-] \mid C \texttt{ ; p} \mid \texttt{if b then } C \texttt{ else p} \mid \texttt{while b do } \{ C \} \mid \ldots$

Contextual Equivalence

Contexts

$C ::= [-] \mid C\ ;\ \mathtt{p} \mid \mathtt{if}\ \mathtt{b}\ \mathtt{then}\ C\ \mathtt{else}\ \mathtt{p} \mid \mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ C\ \} \mid \ldots$

Can one still prove $\mathtt{p} \equiv_o \mathtt{q}$ iff $\mathtt{p} \equiv \mathtt{q}$?

Next Challenge: Programs as part of a Mathematical Theory

Programming language $\hookrightarrow$ Mathematical theory

The latter include *e.g.*

- functions (recall program calculus)
- linear algebra
- relations
- domain theory (theory of computability and beyond)
- ...

Table of Contents

Motivation

My first denotational semantics

Denotational semantics for a while-language

Domain theory

Conclusions

Boolean Terms and their Denotational Semantics

$$\mathtt{b} ::= \mathtt{x} \mid \mathtt{b} \wedge \mathtt{b} \mid \neg \mathtt{b}$$

Terms interpreted as functions $[\![\mathtt{b}]\!] : \mathit{State} \to 2$

Term operations interpreted via the boolean algebra 2

$$[\![\mathtt{x}]\!](\sigma) = \sigma(\mathtt{x})$$
$$[\![\mathtt{b}_1 \wedge \mathtt{b}_2]\!] = (\wedge) \cdot \langle [\![\mathtt{b}_1]\!], [\![\mathtt{b}_2]\!] \rangle$$
$$[\![\neg \mathtt{b}]\!] = (\neg) \cdot [\![\mathtt{b}]\!]$$

Big-step meets Denotational ...

Theorem

$\langle \mathrm{b}, \sigma \rangle \Downarrow v$ *iff* $[\![\mathrm{b}]\!](\sigma) = v$

Proof.

Straightforward induction □

Corollary

$\mathrm{b}_1 \equiv \mathrm{b}_2$ *iff* $\mathrm{b}_1 \equiv_o \mathrm{b}_2$ *iff* $[\![\mathrm{b}_1]\!] = [\![\mathrm{b}_2]\!]$

Profits !

We can now reduce checking equivalence to your favorites . . .

Program calculus and Boolean algebra

Profits !

We can now reduce checking equivalence to your favorites . . .

Program calculus and Boolean algebra

Example

$$
\begin{aligned}
[\![\mathsf{b}_1 \wedge \mathsf{b}_2]\!] &= (\wedge) \cdot \langle [\![\mathsf{b}_1]\!], [\![\mathsf{b}_2]\!] \rangle \\
&= (\wedge) \cdot \mathrm{sw} \cdot \langle [\![\mathsf{b}_1]\!], [\![\mathsf{b}_2]\!] \rangle \\
&= (\wedge) \cdot \langle \pi_2, \pi_1 \rangle \cdot \langle [\![\mathsf{b}_1]\!], [\![\mathsf{b}_2]\!] \rangle \\
&= (\wedge) \cdot \langle \pi_2 \cdot \langle [\![\mathsf{b}_1]\!], [\![\mathsf{b}_2]\!] \rangle, \pi_1 \cdot \langle [\![\mathsf{b}_1]\!], [\![\mathsf{b}_2]\!] \rangle \rangle \\
&= (\wedge) \cdot \langle [\![\mathsf{b}_2]\!], [\![\mathsf{b}_1]\!] \rangle \\
&= [\![\mathsf{b}_2 \wedge \mathsf{b}_1]\!]
\end{aligned}
$$

Exercises

Show $\mathsf{b} \wedge \mathsf{b} \equiv \mathsf{b}$ (via the denotational semantics)

Define a denotational semantics for arithmetic expressions

Show $\mathsf{e}_1 + \mathsf{e}_2 \equiv \mathsf{e}_2 + \mathsf{e}_1$ (via your denotational semantics)

Prove the equivalence $\langle \mathsf{e}, \sigma \rangle \Downarrow v$ iff $[\![\mathsf{e}]\!](\sigma) = v$

Table of Contents

Motivation

My first denotational semantics

Denotational semantics for a while-language

Domain theory

Conclusions

Key Takeaways

Programs interpreted as functions $[\![\mathrm{p}]\!] : \mathit{State}_\perp \to \mathit{State}_\perp$

$\mathit{State}_\perp = \mathit{State} \cup \{\perp\}$ where $\perp$ represents non-termination

Sequential composition is function composition

Programs and their Denotational Semantics

$$\mathtt{p} ::= \mathtt{x} := \mathtt{e} \mid \mathtt{p}\ ;\ \mathtt{p} \mid \texttt{if}\ \mathtt{b}\ \texttt{then}\ \mathtt{p}\ \texttt{else}\ \mathtt{p} \mid \texttt{while}\ \mathtt{b}\ \texttt{do}\ \{\ \mathtt{p}\ \}$$

$$\begin{aligned}
[\![\mathtt{x} := \mathtt{e}]\!] &= \sigma \mapsto \sigma[[\![\mathtt{e}]\!]/\mathtt{x}] \\
[\![\mathtt{p}\ ;\ \mathtt{q}]\!] &= [\![\mathtt{q}]\!] \cdot [\![\mathtt{p}]\!] \\
[\![\texttt{if}\ \mathtt{b}\ \texttt{then}\ \mathtt{p}\ \texttt{else}\ \mathtt{q}]\!] &= [[\![\mathtt{p}]\!], [\![\mathtt{q}]\!]] \cdot \mathrm{dist} \cdot \langle [\![\mathtt{b}]\!], \mathrm{id} \rangle \\
[\![\texttt{while}\ \mathtt{b}\ \texttt{do}\ \{\ \mathtt{p}\ \}]\!] &= \ \ldots
\end{aligned}$$

Big-step and Denotational Semantics

Danger, Will Robinson: no while-loops yet ...

Theorem

$\langle \mathtt{p}, \sigma \rangle \Downarrow \sigma'$ *iff* $[\![\mathtt{p}]\!](\sigma) = \sigma'$

Proof.

Straightforward induction □

Corollary

$\mathtt{p} \equiv \mathtt{q}$ *iff* $\mathtt{p} \equiv_o \mathtt{q}$ *iff* $[\![\mathtt{p}]\!] = [\![\mathtt{q}]\!]$

Profits !

Recall when we had to prove

- $(\texttt{p}\,;\texttt{q})\,;\texttt{r} \equiv \texttt{p}\,;(\texttt{q}\,;\texttt{r})$
- $(\texttt{if b then p else q})\,;\texttt{r} \equiv \texttt{if b then p}\,;\texttt{r else q}\,;\texttt{r}$

with the big-step semantics

Show the same via the denotational semantics

Programs and a Tentative Denotational Semantics

$$\mathtt{p} ::= \mathtt{x} := \mathtt{e} \mid \mathtt{p}\ ;\ \mathtt{p} \mid \texttt{if}\ \mathtt{b}\ \texttt{then}\ \mathtt{p}\ \texttt{else}\ \mathtt{p} \mid \texttt{while}\ \mathtt{b}\ \texttt{do}\ \{\ \mathtt{p}\ \}$$

$$\begin{aligned}
[\![\mathtt{x} := \mathtt{e}]\!] &= \sigma \mapsto \sigma[[\![\mathtt{e}]\!]/\mathtt{x}] \\
[\![\mathtt{p}\ ;\ \mathtt{q}]\!] &= [\![\mathtt{q}]\!] \cdot [\![\mathtt{p}]\!] \\
[\![\texttt{if}\ \mathtt{b}\ \texttt{then}\ \mathtt{p}\ \texttt{else}\ \mathtt{q}]\!] &= [[\![\mathtt{p}]\!], [\![\mathtt{q}]\!]] \cdot \mathrm{dist} \cdot \langle [\![\mathtt{b}]\!], \mathrm{id} \rangle \\
[\![\texttt{while}\ \mathtt{b}\ \texttt{do}\ \{\ \mathtt{p}\ \}]\!] &= [[\![\texttt{while}\ \mathtt{b}\ \texttt{do}\ \{\ \mathtt{p}\ \}]\!] \cdot [\![\mathtt{p}]\!], \mathrm{id}] \cdot \mathrm{dist} \cdot \langle [\![\mathtt{b}]\!], \mathrm{id} \rangle
\end{aligned}$$

Programs and a Tentative Denotational Semantics

$$\mathtt{p} ::= \mathtt{x} := \mathtt{e} \mid \mathtt{p}\ ;\ \mathtt{p} \mid \texttt{if}\ \mathtt{b}\ \texttt{then}\ \mathtt{p}\ \texttt{else}\ \mathtt{p} \mid \texttt{while}\ \mathtt{b}\ \texttt{do}\ \{\ \mathtt{p}\ \}$$

$$\begin{aligned}
[\![\mathtt{x} := \mathtt{e}]\!] &= \sigma \mapsto \sigma[[\![\mathtt{e}]\!]/\mathtt{x}] \\
[\![\mathtt{p}\ ;\ \mathtt{q}]\!] &= [\![\mathtt{q}]\!] \cdot [\![\mathtt{p}]\!] \\
[\![\texttt{if}\ \mathtt{b}\ \texttt{then}\ \mathtt{p}\ \texttt{else}\ \mathtt{q}]\!] &= [[\![\mathtt{p}]\!], [\![\mathtt{q}]\!]] \cdot \mathrm{dist} \cdot \langle [\![\mathtt{b}]\!], \mathrm{id} \rangle \\
[\![\texttt{while}\ \mathtt{b}\ \texttt{do}\ \{\ \mathtt{p}\ \}]\!] &= [[\![\texttt{while}\ \mathtt{b}\ \texttt{do}\ \{\ \mathtt{p}\ \}]\!] \cdot [\![\mathtt{p}]\!], \mathrm{id}] \cdot \mathrm{dist} \cdot \langle [\![\mathtt{b}]\!], \mathrm{id} \rangle
\end{aligned}$$

I'm very clear, Brexit does mean brexit

(Theresa May) `https://www.youtube.com/watch?v=oRDfFJAu6Bo`

Table of Contents

Motivation

My first denotational semantics

Denotational semantics for a while-language

Domain theory

Conclusions

Partially Ordered Set

Definition (Poset)

A set with a reflexive, anti-symmetric, and transitive relation $\leq$

Examples

- $(\mathbb{N},$ the usual order $\leq$ on natural numbers$)$
- $(\mathbb{R},$ the usual order $\leq$ on real numbers$)$
- $(X, =)$ (for any set X)
- $(\mathrm{P}X, \subseteq)$ (for any set X)

Partially Ordered Set

Definition (Poset)

A set with a reflexive, anti-symmetric, and transitive relation $\leq$

Examples

- $(\mathbb{N},$ the usual order $\leq$ on natural numbers$)$
- $(\mathbb{R},$ the usual order $\leq$ on real numbers$)$
- $(X, =)$ (for any set X)
- $(\mathrm{P}X, \subseteq)$ (for any set X)

In our context $x \leq y$ reads as

x less informative than y

New Posets from Old Ones

Addition of a bottom element

If $(X, \leq_X)$ is a poset then $(X_\perp, \leq)$ is a poset when defined as

- $x_1 \leq x_2$ iff $x_1 \leq_X x_2$
- $\perp \leq x$ (for all $x \in X$)

$\perp$ is the least informative element, akin to non-termination

New Posets from Old Ones

Addition of a bottom element

If $(X, \leq_X)$ is a poset then $(X_\perp, \leq)$ is a poset when defined as

- $x_1 \leq x_2$ iff $x_1 \leq_X x_2$
- $\perp \leq x$ (for all $x \in X$)

$\perp$ is the least informative element, akin to non-termination

Example

In what way is $State_\perp$ a poset ?

Data Aggregation

We wish to collect a chain of information

$$x_1 \leq x_2 \leq x_3 \leq \dots$$

into a single datum, denoted by $\bigvee_{i \in \mathbb{N}} x_i$

Data Aggregation

We wish to collect a chain of information

$$x_1 \leq x_2 \leq x_3 \leq \ldots$$

into a single datum, denoted by $\vee_{i \in \mathbb{N}} x_i$

This element should be more informative than any x_j $(j \in \mathbb{N})$ *i.e.*

$$x_j \leq \vee_{i \in \mathbb{N}} x_i$$

. . . and contain no more information than the chain *i.e.*

$$(\forall j \in \mathbb{N}. x_j \leq y) \Longrightarrow \vee_{i \in \mathbb{N}} x_i \leq y$$

Posets + Data Aggregation

Definition (ω-CPO)

A poset with data aggregation as previously described

Examples

- $\mathbb{N}$ is <u>not</u> an ω-CPO but $\mathbb{N} \cup \{\infty\}$ is
- $\mathbb{R}$ is <u>not</u> an ω-CPO but $\mathbb{R} \cup \{\infty\}$ and $[0, 1]$ are
- $(\mathrm{P}X, \subseteq)$ is an ω-CPO for any set X

Posets + Data Aggregation

Definition (ω-CPO)

A poset with data aggregation as previously described

Examples

- $\mathbb{N}$ is <u>not</u> an ω-CPO but $\mathbb{N} \cup \{\infty\}$ is
- $\mathbb{R}$ is <u>not</u> an ω-CPO but $\mathbb{R} \cup \{\infty\}$ and $[0, 1]$ are
- $(\mathrm{P}X, \subseteq)$ is an ω-CPO for any set X

Exercise

Show that $State_\perp$ is an ω-CPO

Maps between ω-CPOs

We wish that maps represent some form of computability ...

... and thus any old map will not do

Maps between ω-CPOs

We wish that maps represent some form of computability ...

... and thus any old map will not do

We will therefore enforce the following laws

$$f(\vee_n x_n) = \vee_n f(x_n) \qquad \text{(continuity)}$$

$$x_1 \leq x_2 \Rightarrow f(x_1) \leq f(x_2) \qquad \text{(monotonicity)}$$

Continuity $\simeq$ Computability ?

What does it mean for $p : X \to \{\bot \leq \top\}$ to be continuous ?

Let $x \in X$ be given by a chain of finite approximations

$$x_1 \leq x_2 \leq x_3 \ldots$$

. . . then deduce that

$$\begin{aligned} p(\vee_n x_n) = \top &\iff \vee_n p(x_n) = \top \\ &\iff \exists n.\, p(x_n) = \top \end{aligned}$$

Continuity $\simeq$ Computability ?

What does it mean for $p : X \to \{\bot \leq \top\}$ to be continuous ?

Let $x \in X$ be given by a chain of finite approximations

$$x_1 \leq x_2 \leq x_3 \ldots$$

. . . then deduce that

$$\begin{aligned} p(\vee_n x_n) = \top &\iff \vee_n p(x_n) = \top \\ &\iff \exists n.\, p(x_n) = \top \end{aligned}$$

i.e. p terminates with $\top$ (true) for x iff p can evaluate a finite approximation of x to $\top$

Continuity $\simeq$ Computability ?

Exercise

Show that $(\mathrm{P}\mathbb{N}, \subseteq)$ is an ω-CPO

Exercise

Is `isInfinite` : $\mathcal{P}(\mathbb{N}) \to \{\bot \leq \top\}$ continuous ?

Continuity $\simeq$ Computability ?

$$[\![\mathtt{x} := \mathtt{e}]\!] = \sigma \mapsto \sigma[[\![\mathtt{e}]\!]/\mathtt{x}]$$

$$[\![\mathtt{p}\,;\,\mathtt{q}]\!] = [\![\mathtt{q}]\!] \cdot [\![\mathtt{p}]\!]$$

$$[\![\mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{q}]\!] = [[\![\mathtt{p}]\!], [\![\mathtt{q}]\!]] \cdot \mathrm{dist} \cdot \langle [\![\mathtt{b}]\!], \mathrm{id} \rangle$$

$$[\![\mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ \mathtt{p}\ \}]\!] = \ \ldots\ldots$$

Are all maps $[\![\mathtt{p}]\!]$ continuous ?

Continuity $\simeq$ Computability ?

$$[\![\mathtt{x} := \mathtt{e}]\!] = \sigma \mapsto \sigma[[\![\mathtt{e}]\!]/\mathtt{x}]$$

$$[\![\mathtt{p}\,;\,\mathtt{q}]\!] = [\![\mathtt{q}]\!] \cdot [\![\mathtt{p}]\!]$$

$$[\![\mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{q}]\!] = [[\![\mathtt{p}]\!], [\![\mathtt{q}]\!]] \cdot \mathrm{dist} \cdot \langle [\![\mathtt{b}]\!], \mathrm{id} \rangle$$

$$[\![\mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ \mathtt{p}\ \}]\!] = \ \ldots\ldots$$

Are all maps $[\![\mathtt{p}]\!]$ continuous ?

Yes, just use program calculus ‘with continuous maps’

Fixpoints for While-loops

Definition

$x \in X$ is a fixpoint of $f : X \to X$ if $f(x) = x$

A notion with powerful applications in diverse fields

- economics (game theory)
- dynamical systems (equilibrium points)
- automata theory
- essentially everywhere ...

Fixpoints for While-loops

Definition

$x \in X$ is a fixpoint of $f : X \to X$ if $f(x) = x$

A notion with powerful applications in diverse fields

- economics (game theory)
- dynamical systems (equilibrium points)
- automata theory
- essentially everywhere ...

While-loops will be fixpoints in our semantics

Fixpoints for While-loops

. . . a fixpoint of which function ?

Fixpoints for While-loops

. . . a fixpoint of which function ?

Recall our previous approach

$$[\![\texttt{while b do \{ p \}}]\!] = [[\![\texttt{while b do \{ p \}}]\!] \cdot [\![\texttt{p}]\!], \mathrm{id}] \cdot \mathrm{dist} \cdot \langle [\![\texttt{b}]\!], \mathrm{id} \rangle$$

It states that $[\![\texttt{while b do \{ p \}}]\!]$ is a fixpoint of

$$k \longmapsto [k \cdot [\![\texttt{p}]\!], \mathrm{id}] \cdot \mathrm{dist} \cdot \langle [\![\texttt{b}]\!], \mathrm{id} \rangle$$

The Least Fixpoint Theorem

Theorem

Every continuous, monotone map $f : X \to X$ has a least fixpoint

$$\mathrm{lfp}\ f = \vee_{n \in \mathbb{N}} f^n(\bot)$$

Exercise

Prove the theorem

The Least Fixpoint Theorem

Theorem

Every continuous, monotone map $f : X \to X$ has a least fixpoint

$$\mathrm{lfp}\ f = \vee_{n \in \mathbb{N}} f^n(\bot)$$

Exercise

Prove the theorem

And finally ...

Programs and a denotational semantics

$$\mathtt{p} ::= \mathtt{x} := \mathtt{e} \mid \mathtt{p}\,;\,\mathtt{p} \mid \mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{p} \mid \mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ \mathtt{p}\ \}$$

$$[\![\mathtt{x} := \mathtt{e}]\!] = \sigma \mapsto \sigma[[\![\mathtt{e}]\!]/\mathtt{x}]$$

$$[\![\mathtt{p}\,;\,\mathtt{q}]\!] = [\![\mathtt{q}]\!] \cdot [\![\mathtt{p}]\!]$$

$$[\![\mathtt{if}\ \mathtt{b}\ \mathtt{then}\ \mathtt{p}\ \mathtt{else}\ \mathtt{q}]\!] = [[\![\mathtt{p}]\!], [\![\mathtt{q}]\!]] \cdot \mathrm{dist} \cdot \langle [\![\mathtt{b}]\!], \mathrm{id} \rangle$$

$$[\![\mathtt{while}\ \mathtt{b}\ \mathtt{do}\ \{\ \mathtt{p}\ \}]\!] = \mathrm{lfp}\left(k \mapsto [k \cdot [\![\mathtt{p}]\!], \mathrm{id}] \cdot \mathrm{dist} \cdot \langle [\![\mathtt{b}]\!], \mathrm{id} \rangle\right)$$

The semantics at Work

Prove the following equivalences

- $\texttt{while}\ \texttt{b}\ \{\texttt{p}\} \equiv \texttt{if}\ \texttt{b}\ \texttt{then}\ \texttt{p}\ ;\ \texttt{while}\ \texttt{b}\ \{\texttt{p}\}\ \texttt{else}\ \texttt{skip}$
- $\texttt{while}\ \texttt{b}\ \{\texttt{p}\}\ ;\ \texttt{q} \equiv \texttt{if}\ \texttt{b}\ \texttt{then}\ \texttt{p}\ ;\ \texttt{while}\ \texttt{b}\ \{\texttt{p}\}\ ;\ \texttt{q}\ \texttt{else}\ \texttt{q}$
- $\texttt{while}\ \texttt{ff}\ \{\texttt{p}\}\ ;\ \texttt{q} \equiv \texttt{q}$
- $\texttt{while}\ \texttt{tt}\ \{\texttt{p}\} \equiv \texttt{while}\ \texttt{tt}\ \{\texttt{q}\}$

The semantics at Work

Prove the following equivalences

- $\texttt{while}\ \texttt{b}\ \{\texttt{p}\} \equiv \texttt{if}\ \texttt{b}\ \texttt{then}\ \texttt{p}\ ;\ \texttt{while}\ \texttt{b}\ \{\texttt{p}\}\ \texttt{else}\ \texttt{skip}$
- $\texttt{while}\ \texttt{b}\ \{\texttt{p}\}\ ;\ \texttt{q} \equiv \texttt{if}\ \texttt{b}\ \texttt{then}\ \texttt{p}\ ;\ \texttt{while}\ \texttt{b}\ \{\texttt{p}\}\ ;\ \texttt{q}\ \texttt{else}\ \texttt{q}$
- $\texttt{while}\ \texttt{ff}\ \{\texttt{p}\}\ ;\ \texttt{q} \equiv \texttt{q}$
- $\texttt{while}\ \texttt{tt}\ \{\texttt{p}\} \equiv \texttt{while}\ \texttt{tt}\ \{\texttt{q}\}$

Prove the following implication

$$[\![\texttt{p}]\!] = [\![\texttt{q}]\!] \Longrightarrow \text{for all contexts } C.\ [\![C[\texttt{p}]]\!] = [\![C[\texttt{q}]]\!]$$

The relation between big-step and denotational semantics

Theorem

$\langle \mathrm{p}, \sigma \rangle \Downarrow \sigma'$ *iff* $[\![\mathrm{p}]\!](\sigma) = \sigma'$

Proof

Yet again ... induction

Corollary

$\mathrm{p} \equiv_o \mathrm{q}$ *iff* $[\![\mathrm{p}]\!] = [\![\mathrm{q}]\!]$

Corollary (Holy grail)

$[\![\mathrm{p}]\!] = [\![\mathrm{q}]\!]$ *iff* $\forall C.\, [\![C[\mathrm{p}]]\!] = [\![C[\mathrm{q}]]\!]$ *iff* $\mathrm{p} \equiv \mathrm{q}$

Table of Contents

Motivation

My first denotational semantics

Denotational semantics for a while-language

Domain theory

Conclusions

Conclusions

We (very briefly) studied denotational semantics

It typically requires more investment than operational counterparts

. . . but this usually pays off

Numerous extensions to emerging programming languages

. . . quantum, probabilistic, hybrid, stochastic . . .

Further details in *e.g.* [Rey98, Chapter 2] and [Win93, Chapter 5].


John C Reynolds, *Theories of programming languages*, Cambridge University Press, 1998.


Glynn Winskel, *The formal semantics of programming languages - an introduction*, Foundation of computing series, MIT Press, 1993.