MPhil ACS, module L16

Semantics of HOT Languages

Andrew Pitts

UNIVERSITY OF CAMBRIDGE

Computer Laboratory

Lent Term 2010

First part of the course follows

AMP, "Operational Semantics & Program Equivalence" — a chapter in :

G. Barthe et al, "Applied Semantics", Lecture Notes in Computer Science, Vol. 2395 (Springer, 2002), pages 378-412.

N.B. NO LECTURE THURSDAY 21 JAN

Introduction:
Programming Language Semantics

Theoretical Computer Science

- Foundations
  Mathematical theory of computation
- Algorithms and complexity
  What can be computed in practice?
- Semantics
  What is the structure of computation?

Programming Language Semantics

What's it for?

What is it?

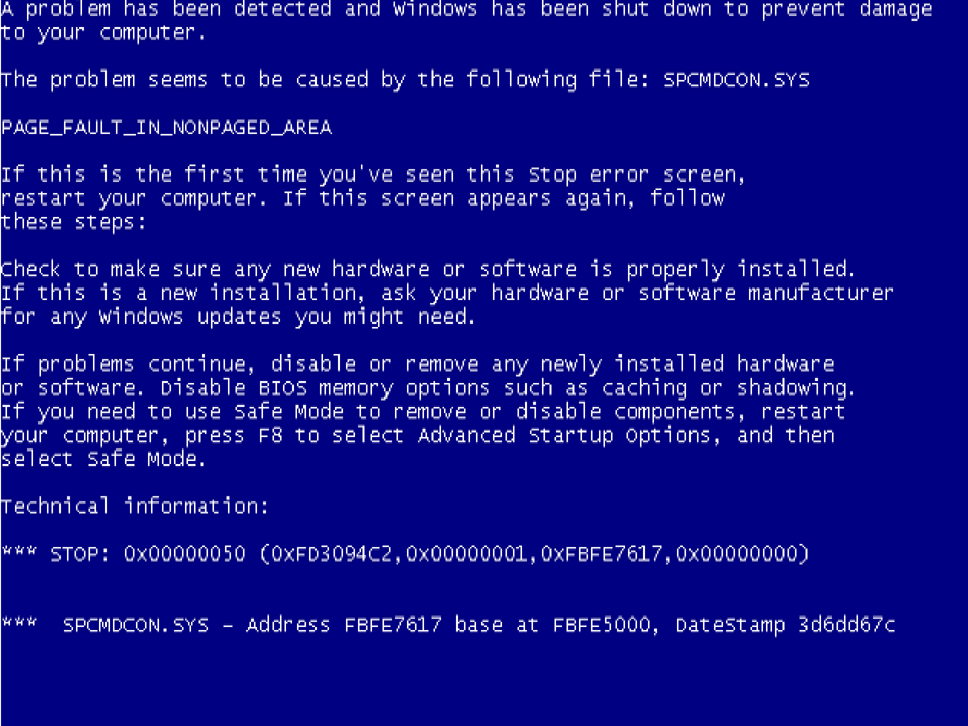
What's there to do?

Programming Language Semantics

What's it for?

What is it?

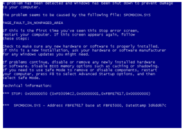
What's there to do?



```
A problem has been detected and Windows has been shut down to prevent damage
to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)


***  SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c
```

Software verification

Software failures range from

frequent-and-annoying: 

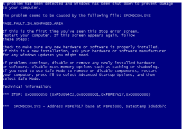
```
A problem has been detected and Windows has been shut down to prevent damage
to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)


***  SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c
```

to life-threatening-and-catastrophic:

a single line of code causes Ariane 5 to abort

Software verification

Software failures range from

frequent-and-annoying: 

```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)


***  SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c
```

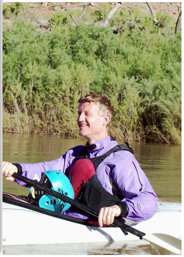
to life-threatening-and-catastrophic:

- Software is at the core of the world's infrastructure.
- No software without programming languages.
- Programming language semantics is crucial for rigorous/systematic software correctness.

Semantics: what's it for?

- Program verification.
- Implementation of existing programming languages.
  e.g. semantics-preserving compiler optimizations

> "Higher-order languages, such as Haskell, encourage the programmer to build abstractions by composing functions. A good compiler must inline many of these calls to recover an efficiently executable program. In principle, inlining is dead simple: just replace the call of a function by an instance of its body. But any compiler-writer will tell you that inlining is a black art, full of delicate compromises that work together to give good performance without unnecessary code bloat."
>
> *Simon Peyton Jones*



Semantics: what's it for?

- Program verification.
- Implementation of existing programming languages.
- Design of new progamming languages.

  e.g. web programming

"A typical, modern web program involves many "tiers": part of the program runs in the web browser, part runs on a web server, and part runs in back-end systems such as a relational database. To create such a program, the programmer must master a myriad of languages: the logic is written in a mixture of Java, Python, and Perl; the presentation in HTML; the GUI behavior in Javascript; and the queries are written in SQL or XQuery. There is no easy way to link these, for example, to be sure that an HTML form or an SQL query produces the type of data that the Java code expects. This problem is called the impedance mismatch problem."

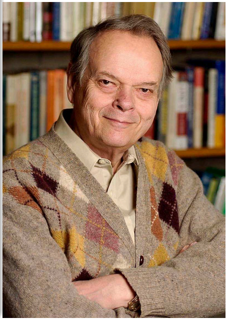
Phil Wadler

Semantics: what's it for?

- Program verification.
- Implementation of existing programming languages.
- Design of new progamming languages.

"Why is it so hard to design a good programming language? Naively, one might expect that a straightforward extension of the conventional notation of science and mathematics should provide a completely adequate programming language. But the history of language design has distroyed this illusion.
"The truth of the matter is that putting languages together is a very tricky business. When one attempts to combine language concepts, unexpected and counterintuitive interactions arise. At this point, even the most experienced designer's intuition must be butressed by a rigorous definition of what the language means.
"Of course, this is what programming language semantics is all about."

John Reynolds, 1990



Programming Language Semantics

What's it for?

What is it?

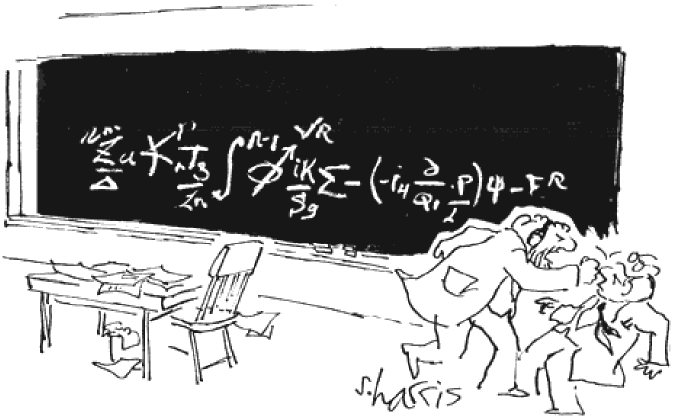
What's there to do?

Styles of semantics

- Program logics
- Denotational semantics
- Operational semantics

Program logic

It's all about proofs of program properties.



"You want proof? I'll give you proof!"

Program logic

It's all about proofs of program properties.

E.g. Floyd-Hoare logic for partial correctness

$$\vdash \{P\}\, C\, \{Q\}$$

Intended meaning:

> "whenever C is executed in a state satisfying pre-condition P and if the execution of C terminates, then the state in which C's execution terminates satisfies post-condition Q"

Program logic

It's all about proofs of program properties.

E.g. Floyd-Hoare logic for partial correctness

$$\vdash \{P\}\, C\, \{Q\}$$

The collection of valid partial correctness assertions is inductively defined by axioms and rules, such as

$$\frac{\vdash \{P \& B\}\, C\, \{P\}}{\vdash \{P\}\, \mathtt{while}(B)\, C\, \{P \& \neg B\}}$$

Program logic

It's all about proofs of program properties.

E.g. Floyd-Hoare-Jones logic for partial correctness

$$R, G \vdash \{P\}\, C\, \{Q\}$$

The collection of valid partial correctness assertions is inductively defined by axioms and rules, such as

$$\frac{\begin{array}{cc} R_1, G_1 \vdash \{P_1\}\, C_1\, \{Q_1\} & G_1 \subseteq R_2 \\ R_2, G_2 \vdash \{P_2\}\, C_2\, \{Q_2\} & G_2 \subseteq R_1 \end{array}}{R_1 \& R_2, G_1 \vee G_2 \vdash \{P_1 \& P_2\}\, C_1 \parallel C_2\, \{Q_1 \& Q_2\}}$$

Program logic

It's all about proofs of program properties.

E.g. Floyd-Hoare-Jones logic for partial correctness

$$R, G \vdash \{P\}\, C\, \{Q\}$$

The collection of valid partial correctness assertions is inductively defined by axioms and rules, such as

$$\frac{\begin{array}{cc} R_1, G_1 \vdash \{P_1\}\, C_1\, \{Q_1\} & G_1 \subseteq R_2 \\ R_2, G_2 \vdash \{P_2\}\, C_2\, \{Q_2\} & G_2 \subseteq R_1 \end{array}}{R_1 \& R_2, G_1 \vee G_2 \vdash \{P_1 \& P_2\}\, C_1 \parallel C_2\, \{Q_1 \& Q_2\}}$$

Where do such "programming laws" come from?

Styles of semantics

- Program logics
- Denotational semantics
- Operational semantics

Denotational semantics

- Each program phrase P is given a denotation $[\![P]\!]$—a mathematical object representing the contribution of P to the meaning of *any* complete program in which it occurs.
- The denotation of a phrase is a function of the denotations of its subphrases—one says that the semantics is compositional.

Denotational semantics

E.g. if

partial functions

$[\![\textbf{statement}]\!] \in \textit{State} \rightharpoonup \textit{State}$

$[\![\textbf{boolean expression}]\!] \in \textit{State} \rightharpoonup \{\texttt{true}, \texttt{false}\}$

then

Denotational semantics

E.g. if

$[\![\textbf{statement}]\!] \in \textit{State} \rightharpoonup \textit{State}$

$[\![\textbf{boolean expression}]\!] \in \textit{State} \rightharpoonup \{\texttt{true}, \texttt{false}\}$

then

$$[\![\texttt{if}\,(B)\,C\,\texttt{else}\,C']\!](s) \equiv cond([\![B]\!](s), [\![C]\!](s), [\![C']\!](s))$$

where

$$cond(b, s, s') = \begin{cases} s & \text{if } b = \texttt{true} \\ s' & \text{if } b = \texttt{false} \end{cases}$$

Denotational semantics

E.g. if

$[\![\textbf{statement}]\!] \in \textit{State} \rightharpoonup \textit{State}$

$[\![\textbf{boolean expression}]\!] \in \textit{State} \rightharpoonup \{\texttt{true}, \texttt{false}\}$

then

$$[\![\texttt{while}(B)\ C]\!] = \mathit{lfp}(\Phi_{[\![B]\!],[\![C]\!]})$$

where

$\Phi_{[\![B]\!],[\![C]\!]} \in (\textit{State} \rightharpoonup \textit{State}) \to (\textit{State} \rightharpoonup \textit{State})$

is

$f \mapsto (s \mapsto \mathit{cond}([\![B]\!](s), f([\![C]\!](s)), s))$

Denotational semantics

E.g. if

$[\![\textbf{statement}]\!] \in \textit{State} \rightharpoonup \textit{State}$

$[\![\textbf{boolean expression}]\!] \in \textit{State} \rightharpoonup \{\texttt{true}, \texttt{false}\}$

then

$$[\![\texttt{while}(B)\ C]\!] = \overset{\text{least fixed point:}}{\mathit{lfp}}(\Phi_{[\![B]\!],[\![C]\!]})$$

- $\Phi_{[\![B]\!],[\![C]\!]}(\mathit{lfp}) = \mathit{lfp}$
- $\Phi_{[\![B]\!],[\![C]\!]}(f) = f \implies \mathit{lfp} \subseteq f$

Domain equations

For example:

$$\begin{aligned} E &= S \rightharpoonup (\mathbb{N} \times S) \\ S &= \mathbb{N} \rightharpoonup E \end{aligned}$$

Domain equations

For example:

denotations of numerical expressions with effects on state

$$E = S \rightharpoonup (\mathbb{N} \times S)$$
$$S = \mathbb{N} \rightharpoonup E$$

States storing a mutable method that applies to numbers

Domain equations

For example:

$$\begin{aligned} E &= S \rightharpoonup (\mathbb{N} \times S) \\ S &= \mathbb{N} \rightharpoonup E \end{aligned}$$

So E has to satisfy

$$E = (\mathbb{N} \rightharpoonup E) \rightharpoonup (\mathbb{N} \times (\mathbb{N} \rightharpoonup E))$$

Domain equations

For example:

$$\begin{aligned} E &= S \rightharpoonup (\mathbb{N} \times S) \\ S &= \mathbb{N} \rightharpoonup E \end{aligned}$$

So E has to satisfy

$$E \underset{\uparrow}{=} (\mathbb{N} \rightharpoonup E) \rightharpoonup (\mathbb{N} \times (\mathbb{N} \rightharpoonup E))$$

Cantor: there are no such <u>sets</u> E.

$$\mathrm{card}(\mathrm{RHS}) \geq 2^{\mathrm{card}(\mathrm{LHS})}$$

$$> \mathrm{card}(\mathrm{LHS})$$

Domain equations

For example:

$$\begin{aligned} E &= S \rightharpoonup (\mathbb{N} \times S) \\ S &= \mathbb{N} \rightharpoonup E \end{aligned}$$

So E has to satisfy

$$E = (\mathbb{N} \rightharpoonup E) \rightharpoonup (\mathbb{N} \times (\mathbb{N} \rightharpoonup E))$$

Cantor: there are no such sets E.

So we have to solve such equations in categories of mathematical structure other than sets.

Theoretical Computer Science

- Foundations
  Mathematical theory of computation
- Algorithms and complexity
  What can be computed in practice?
- Semantics
  What is the structure of computation?

Category Theory = maths of structure via transformation (morphisms)

very important in development of denotational semantics

Styles of semantics

- Program logics
- Denotational semantics
- Operational semantics

Operational semantics

E.g. transition relation between abstract machine configurations

$$\langle s, C\rangle \rightarrow \langle s', C'\rangle$$

configuration = state s + program phrase C

Operational semantics

E.g. transition relation between abstract machine configurations

$$\langle s\,, C\rangle \to \langle s'\,, C'\rangle$$

Operational semantics

E.g. transition relation between abstract machine configurations

$$\langle s, C\rangle \to \langle s', C'\rangle$$

The collection of valid transitions is inductively defined by axioms and rules, such as

$$\langle s, \mathtt{while}(B)\, C\rangle \to \langle s, \mathtt{if}(B)\,\{C\,;\mathtt{while}(B)\, C\}\rangle$$

and

$$\frac{\langle s, B\rangle \to \langle s', B'\rangle}{\langle s, \mathtt{if}(B)\, C\rangle \to \langle s', \mathtt{if}(B')\, C\rangle}$$

Styles of semantics

1. Program logics
2. Denotational semantics
3. Operational semantics

This course: mainly 3, some 2, no 1.