### **Denotational Semantics**

10 lectures for Part II CST 2017/18

Marcelo Fiore

Course web page:

http://www.cl.cam.ac.uk/teaching/1718/DenotSem/

## Topic 1

Introduction

#### What is this course about?

• General area.

*Formal methods*: Mathematical techniques for the specification, development, and verification of software and hardware systems.

• Specific area.

*Formal semantics*: Mathematical theories for ascribing meanings to computer languages.

#### Why do we care?

- Rigour.
  - ... specification of programming languages
  - ... justification of program transformations
- Insight.
  - ... generalisations of notions computability
  - ... higher-order functions
  - ... data structures

- Feedback into language design.
  - ... continuations
  - ... monads
- Reasoning principles.
  - ... Scott induction
  - ... Logical relations
  - ... Co-induction

#### **Operational.**

Meanings for program phrases defined in terms of the *steps of computation* they can take during program execution.

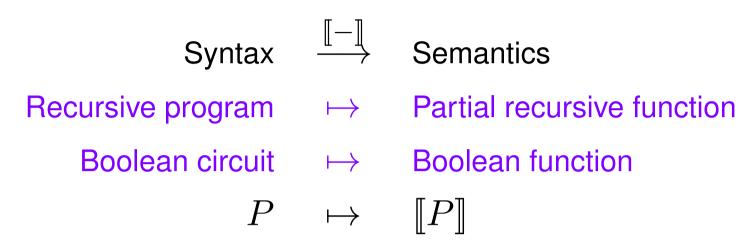
#### Axiomatic.

Meanings for program phrases defined indirectly via the *axioms and rules* of some logic of program properties.

#### **Denotational**.

Concerned with giving *mathematical models* of programming languages. Meanings for program phrases defined abstractly as elements of some suitable mathematical structure.

#### **Basic idea of denotational semantics**



#### **Concerns:**

- Abstract models (*i.e.* implementation/machine independent).

   — Lectures 2, 3 and 4.
- Compositionality.
  - $\rightsquigarrow$  Lectures 5 and 6.
- Relationship to computation (*e.g.* operational semantics).
   ~> Lectures 7 and 8.

# Characteristic features of a denotational semantics

- Each phrase (= part of a program), 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 determined just by the denotations of its subphrases (one says that the semantics is compositional).

IMP<sup>-</sup> syntax

Arithmetic expressions

 $A \in \mathbf{Aexp} ::= \underline{n} \mid L \mid A + A \mid \dots$ 

where n ranges over *integers* and L over a specified set of *locations* L

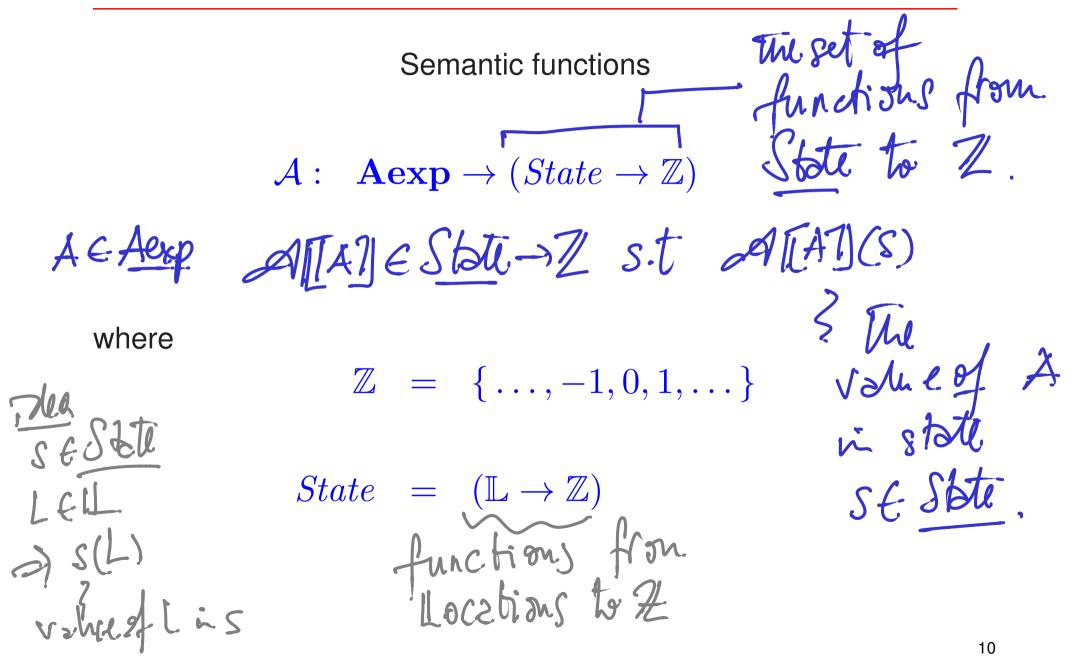
Boolean expressions

 $B \in \mathbf{Bexp} \quad ::= \quad \mathbf{true} \mid \mathbf{false} \mid A = A \mid \dots \\ \mid \quad \neg B \mid \dots$ 

Commands

 $C \in \mathbf{Comm} \quad ::= \quad \mathbf{skip} \quad | \quad L := A \quad | \quad C; C$  $| \quad \mathbf{if} \ B \mathbf{then} \ C \mathbf{else} \ C$ 

**Basic example of denotational semantics (II)** 



Semantic functions

$$\mathcal{A}: \quad \mathbf{Aexp} \to (State \to \mathbb{Z})$$
$$\mathcal{B}: \quad \mathbf{Bexp} \to (State \to \mathbb{B})$$

where

$$\mathbb{Z} = \{\dots, -1, 0, 1, \dots\}$$
$$\mathbb{B} = \{true, false\}$$
$$State = (\mathbb{L} \to \mathbb{Z})$$

Semantic functions  

$$C \in Comm \qquad C \parallel C \parallel \sim State \land Taus for mer.$$

$$A : A \exp \rightarrow (State \rightarrow \mathbb{Z})$$

$$B : B \exp \rightarrow (State \rightarrow \mathbb{B})$$

$$C : Comm \rightarrow (State \rightarrow State)$$
where
$$\mathbb{Z} = \{\dots, -1, 0, 1, \dots\} \quad State \quad to$$

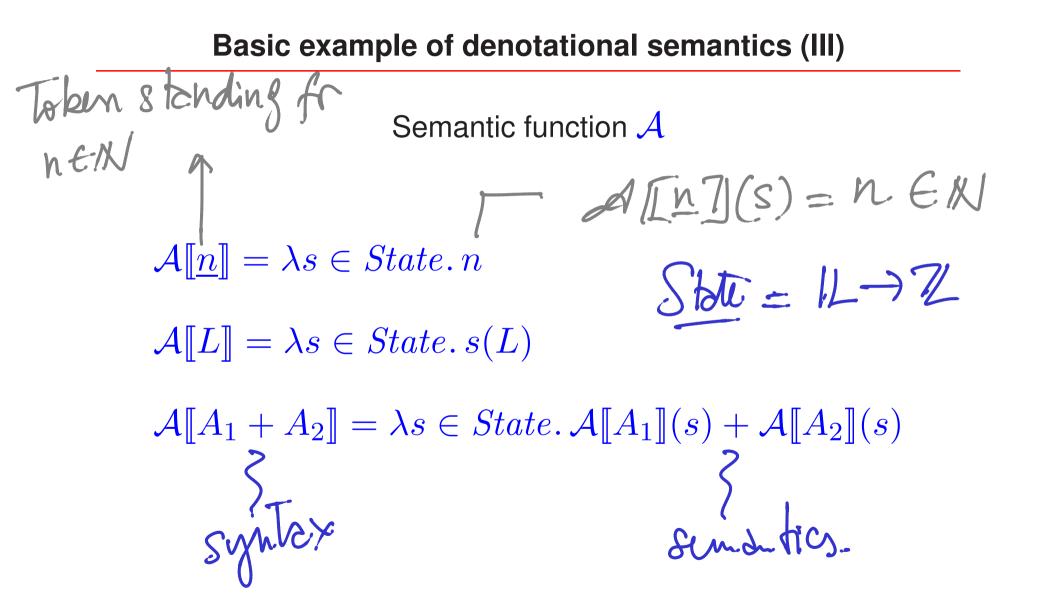
$$\mathbb{Z} = \{true, false\}$$

$$B = \{true, false\}$$

$$State = (\mathbb{L} \rightarrow \mathbb{Z})$$

$$mdefined \quad means$$

$$C \quad bopp.$$



Semantic function  $\mathcal{B}$ 

 $\mathcal{B}\llbracket \mathbf{true} \rrbracket = \lambda s \in State. true$  $\mathcal{B}\llbracket \mathbf{false} \rrbracket = \lambda s \in State. false$  $\mathcal{B}\llbracket A_1 = A_2 \rrbracket = \lambda s \in State. eq \left(\mathcal{A}\llbracket A_1 \rrbracket(s), \mathcal{A}\llbracket A_2 \rrbracket(s)\right)$  $\text{where } eq(a, a') = \begin{cases} true & \text{if } a = a' \\ false & \text{if } a \neq a' \end{cases}$ 

#### **Basic example of denotational semantics (V)**

Semantic function 
$$C$$
  
 $E(State - State)$   
 $[skip] = \lambda s \in State. s$   $M$  identity  
function

**NB:** From now on the names of semantic functions are omitted!

#### A simple example of compositionality

Given partial functions  $\llbracket C \rrbracket$ ,  $\llbracket C' \rrbracket$ : *State*  $\rightarrow$  *State* and a function  $\llbracket B \rrbracket$ : *State*  $\rightarrow$  {*true*, *false*}, we can define  $Composition \Im A \cup Composition \Im A \cup Comp$ 

where

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

#### **Basic example of denotational semantics (VI)**

Semantic function  $\mathcal{C}$ 

$$\begin{bmatrix} L := A \end{bmatrix} = \lambda s \in State. \ \lambda \ell \in \mathbb{L}. \ if \left( \ell = L, \llbracket A \rrbracket(s), s(\ell) \right)$$
  
by Compositionality  
from in This of  

$$\begin{bmatrix} A \end{bmatrix}.$$

Denotation of sequential composition C; C' of two commands

$$\llbracket C; C' \rrbracket = \llbracket C' \rrbracket \circ \llbracket C \rrbracket = \lambda s \in State. \llbracket C' \rrbracket (\llbracket C \rrbracket (s))$$

given by composition of the partial functions from states to states  $\llbracket C \rrbracket, \llbracket C' \rrbracket : State \rightarrow State$  which are the denotations of the commands.

Cf. operational semantics of sequential composition:

$$\frac{C, s \Downarrow s' \quad C', s' \Downarrow s''}{C; C', s \Downarrow s''} \cdot C, s \Downarrow s'' \cdot C, s \Downarrow s'' \cdot C, s \Downarrow s' \Leftrightarrow ICM(s) = s'$$

 $[while B do C] \in (State \rightarrow State)$ Inhle B do CJ(S) = .... [By(S) --- [CJ(S) ---[Imlife B do C7](S) Tubile B do CJ (TCIS), S) = if(IBV(S)),\* Sadefinition in terms of streff X idea: \* not compositional while B do C. = If B Then (C; while BdoC) else skip 17