G54FOP: Lecture 16 Denotational Semantics and Domain Theory II

Henrik Nilsson

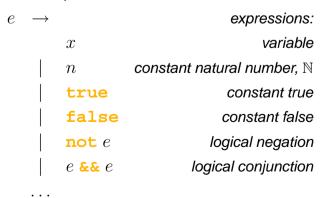
University of Nottingham, UK

This Lecture

- Denotational semantics for small imperative language.
- Introduction to semantics of loops and recursion.

Imperative Language (1)

Syntax of expressions:



Imperative Language (2)

e	\rightarrow		expressions:
	• • •		
		e + e	addition
		e - e	subtraction
		<i>e</i> = <i>e</i>	numeric equality test
		e < e	numeric less than test

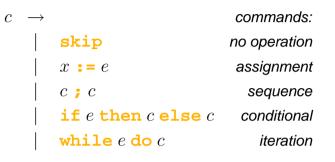
G54FOP: Lecture 16 – p.3/8

G54FOP: Lecture 16 - p.1/8

G54FOP: Lecture 16 - p.2/8

Imperative Language (3)

Syntax of commands:



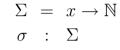
G54FOP: Lecture 16 - p.5/8

G54FOP: Lecture 16 - p.7/8

Semantics of Expressions (1)

We take the *semantic domain* to be \mathbb{N} for simplicity.

We need a way to give meaning to *variables*. A *store* maps a variable name to its value:



Semantics of Expressions (2)

We then need two *semantic functions*, one for expressions (have no side effects in this language), one for commands.

Starting with the one for expressions:

$$\mathbf{E}\llbracket \cdot \rrbracket : e \to (\Sigma \to \mathbb{N})$$

(Note:
$$e \to (\Sigma \to \mathbb{N}) = e \to \Sigma \to \mathbb{N}$$
 etc.)

(Definition on whiteboard)

Semantics of Commands

A command is executed for its *effects*: given a state, executing a command results in a new state. A command is a *state transformer*.

In our case, the state comprises only the store:

 $\Sigma = x \to \mathbb{N}$

Thus, type of state transformer: $\Sigma \rightarrow \Sigma$.

Semantic function for commands:

 $C[\![\cdot]\!] : c \to (\Sigma \to \Sigma)$ [Not correct yet!]

(Definition on whiteboard)

G54FOP: Lecture 16 - p.6/8