Introduction to Semantics

Hanne Riis Nielson

Informatics and Mathematical Modelling

Technical University of Denmark

Syntactic categories and abstract syntax

- numerals: $n \in \mathbf{Num}$
- variables: $x \in \mathbf{Var}$
- arithmetic expressions: $a \in \mathbf{Aexp}$

$$a \quad ::= \quad n \mid x \mid a_1 + a_2 \mid a_1 \star a_2 \mid a_1 - a_2$$

- boolean expressions: $b \in \mathbf{Bexp}$

$$b \quad ::= \quad \texttt{true} \mid \texttt{false} \mid a_1 = a_2 \mid a_1 \leq a_2 \mid \neg b \mid b_1 \wedge b_2$$

- statements: $S \in \mathbf{Stm}$

$$\begin{array}{lll} S & ::= & x := a \mid \texttt{skip} \mid S_1 ; S_2 \mid \texttt{if}\ b\ \texttt{then}\ S_1\ \texttt{else}\ S_2 \\ & \mid & \texttt{while}\ b\ \texttt{do}\ S \mid \texttt{repeat}\ S\ \texttt{until}\ b \end{array}$$

Add parentheses to disambiguate the abstract syntax!

Semantic categories

- Natural numbers: $\mathbf{N} = \{0, 1, 2, \cdots\}$
- Truth values: $\mathbf{T} = \{\mathsf{tt}, \mathsf{ff}\}$
- States: $\mathbf{State} = \mathbf{Var} \to \mathbf{N}$ alternatives: $\mathbf{State} = (\mathbf{Var} \times \mathbf{N})^*$

  $\mathbf{State} = \mathbf{Var}^* \times \mathbf{N}^*$

  Operations on states:

  - lookup in a state: $s\ x$
  - update a state: $s' = s[y \mapsto n]$

$$s'\ x = \begin{cases} s\ x & \text{if } x \neq y \\ n & \text{if } x = y \end{cases}$$

Semantic functions

- numerals: $\mathcal{N} : \mathbf{Num} \to \mathbf{N}$
- variables: $s \in \mathbf{State} = \mathbf{Var} \to \mathbf{N}$
- arithmetic expressions: $\mathcal{A} : \mathbf{Aexp} \to (\mathbf{State} \to \mathbf{N})$
- boolean expressions: $\mathcal{B} : \mathbf{Bexp} \to (\mathbf{State} \to \mathbf{T})$
- statements: $\mathcal{S} : \mathbf{Stm} \to (\mathbf{State} \hookrightarrow \mathbf{State})$

The semantics of statements are partial functions — programs may loop!

Arithmetic expressions

$\mathcal{A} : \mathbf{Aexp} \to (\mathbf{State} \to \mathbf{N})$

$$
\begin{aligned}
\mathcal{A}[\![n]\!]s &= \mathcal{N}[\![n]\!] \\
\mathcal{A}[\![x]\!]s &= s\ x \\
\mathcal{A}[\![a_1 + a_2]\!]s &= \mathcal{A}[\![a_1]\!]s + \mathcal{A}[\![a_2]\!]s \\
\mathcal{A}[\![a_1 \star a_2]\!]s &= \mathcal{A}[\![a_1]\!]s \star \mathcal{A}[\![a_2]\!]s \\
\mathcal{A}[\![a_1 - a_2]\!]s &= \mathcal{A}[\![a_1]\!]s - \mathcal{A}[\![a_2]\!]s
\end{aligned}
$$

Boolean expressions

$\mathcal{B} : \mathbf{Bexp} \to (\mathbf{State} \to \mathbf{T})$

$$\mathcal{B}[\![\texttt{true}]\!]s = \mathbf{tt}$$

$$\mathcal{B}[\![\texttt{false}]\!]s = \mathbf{ff}$$

$$\mathcal{B}[\![a_1 = a_2]\!]s = \begin{cases} \mathbf{tt} & \text{if } \mathcal{A}[\![a_1]\!]s = \mathcal{A}[\![a_2]\!]s \\ \mathbf{ff} & \text{if } \mathcal{A}[\![a_1]\!]s \neq \mathcal{A}[\![a_2]\!]s \end{cases}$$

$$\mathcal{B}[\![a_1 \leq a_2]\!]s = \begin{cases} \mathbf{tt} & \text{if } \mathcal{A}[\![a_1]\!]s \leq \mathcal{A}[\![a_2]\!]s \\ \mathbf{ff} & \text{if } \mathcal{A}[\![a_1]\!]s > \mathcal{A}[\![a_2]\!]s \end{cases}$$

$$\mathcal{B}[\![\neg\, b]\!]s = \begin{cases} \mathbf{tt} & \text{if } \mathcal{B}[\![b]\!]s = \mathbf{ff} \\ \mathbf{ff} & \text{if } \mathcal{B}[\![b]\!]s = \mathbf{tt} \end{cases}$$

$$\mathcal{B}[\![b_1 \wedge b_2]\!]s = \begin{cases} \mathbf{tt} & \text{if } \mathcal{B}[\![b_1]\!]s = \mathbf{tt} \text{ and } \mathcal{B}[\![b_2]\!]s = \mathbf{tt} \\ \mathbf{ff} & \text{if } \mathcal{B}[\![b_1]\!]s = \mathbf{ff} \text{ or } \mathcal{B}[\![b_2]\!]s = \mathbf{ff} \end{cases}$$

Compositional definitions

- The syntactic category is specified by an abstract syntax defining
  - the basis elements and
  - the composite elements.

  The composite elements have a unique decomposition into their immediate constituents.

- The semantics is defined by compositional definitions of a function:
  - there is a semantic clause for each of the basis elements of the syntactic category, and
  - there is a semantic clause for each of the methods for constructing composite elements; the clause for a composite element is defined in terms of the semantics of the immediate constituents of the element.

Proof principle: Structural induction

To prove a property of all the elements of the syntactic category do the following:

- Prove that the property holds for all the basis elements of the syntactic category.
- Prove that the property holds for all the composite elements of the syntactic category: Assume that the property holds for all the immediate constituents of the element — this is called the induction hypothesis — and prove that it also holds for the element itself.

Statements

$\mathcal{S} : \mathbf{Stm} \rightarrow (\mathbf{State} \hookrightarrow \mathbf{State})$

- Operational semantics

  specified by transition systems:

  - structural operational semantics

    focus on the individual steps of the computation

  - natural semantics

    focus on the overall result of the computation

- Denotational semantics

  specified by mathematical functions

Operational Semantics

— how to execute the program

$$y := 1;\ \texttt{while}\ \neg(x = 1)\ \texttt{do}\ (y := x \star y; x := x - 1)$$

First we assign 1 to y, the we test whether x is 1 or not. If it is then we stop and otherwise we update y to be the product of x and the previous value of y and then we decrement x by one. Now we test whether the new value of x is 1 or not . . .

The semantics is an abstraction of how the program is executed on a machine.

Denotational Semantics — the function computed by the program

$$y := 1;\ \mathtt{while}\ \neg(x = 1)\ \mathtt{do}\ (y := x \star y; x := x - 1)$$

The program computes a partial function from states to states: the final state will equal the initial state except that the value of x is 1 and the value of y is the factorial of the value of x in the initial state.

The semantics abstracts away from how programs are executed.