

## 2D1455 Foundations of Object-Orientation

Hemtenta 2006-05-17  
Model Answers

### Question 1.

(i) Let  $C = \{c_1, c_2, \dots\}$  be an infinite collection of class names, let  $C^+ = \{int, bool, c_1, c_2, \dots\}$   
Define  $C^{Array}$  inductively by

- (a)  $C^+ \subseteq C^{Array}$
- (b) If  $c \in C^{Array}$  then  $array(c) \in C^{Array}$ ,
- (c)  $C^{Array}$  is the smallest set satisfying (a) and (b).

(ii) Instance (attribute) and temporary variables, as well as method parameters and return types, will now range over  $C^{Array}$  instead of  $C^+$ .

So we have  $IVar(c)_d$  for  $c \in C$ ,  $d \in C^{Array}$  and  $TVar_d$  for  $d \in C^{Array}$ , and  $MName_{d_0, d_1, \dots, d_n}^c$  for  $n \geq 0$ ,  $c \in C$  and  $d_0, d_1, \dots, d_n \in C^{Array}$

The set  $Exp(c)_d$  of expressions for class  $c \in C$  of type  $d \in C^{Array}$  extends the basic definition in the notes by:

$$\begin{aligned} e_d^c &::= x_d^c \mid \dots \mid \\ &length( e_{array(d')}^c ) \mid \text{if } d = int \\ &e_{array(d)}^c [ f_{int}^c ] \mid \dots \end{aligned}$$

(iii) The set  $SExp(c)_d$  of expressions with side effects for class  $c \in C$  of type  $d \in C^{Array}$  extends the basic definition in the notes by:

$$\begin{aligned} s_d^c &::= e_d^c \mid \dots \mid \\ &new_{array(d')} ( e_{int}^c ) \mid \text{if } d = array(d') \text{ for some } d' \in C^{Array} \end{aligned}$$

(iv) The set  $Stat(c)$  of statements for a class  $c \in C$  extends the definition in the notes by

$$\begin{aligned} s^c &::= x_d^c = s_d^c \mid \dots \mid \\ &x_{array(d)}^c [ e_{int}^c ] = s_d^c \mid \\ &u_{array(d)} [ e_{int}^c ] = s_d^c \end{aligned}$$

(v) For a given class  $c \in C$  let  $O^c$  be an infinite set of object names of class  $c$ ,  $\beta^c \in O^c$ . Let  $O^\perp = O^c \cup \{\perp_c\}$  Let  $(O^\perp)^*$  be the set of all finite sequences (vectors or lists) of members of  $O^\perp$

For the given  $C$ ,  $IVar$  and  $TVar$  we define

$$\begin{aligned} State &= \prod_{c \in C} p^{fn}(O^c) \times \\ &\prod_{c \in C, d \in C^+} O^c \times IVar(c)_d \rightarrow O^\perp^d \times \\ &\prod_{c \in C, d \in C^{Array}} O^c \times IVar(c)_{array(d)} \rightarrow (O^\perp^d)^* \times \\ &\prod_{d \in C^+} TVar_d \rightarrow O^\perp^d \times \\ &\prod_{d \in C^{Array}} TVar_{array(d)} \rightarrow (O^\perp^d)^* \end{aligned}$$

Then a state is a 5-tuple

$$\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5)$$

where  $\sigma_1$  is the set of currently active objects,  $\sigma_2$  is a binding of non-array attributes to values (integers, booleans and objects),  $\sigma_3$  is a binding of array attributes to finite sequences of values (integers, booleans, objects and arrays),  $\sigma_4$  is a binding of temporary variables to values (integers, booleans and objects), and  $\sigma_5$  is a binding of temporary array variables to finite sequences of values (integers, booleans, objects and arrays).

The value  $\perp_{array(d)}$  will be assigned to an instance variable  $x_{array(d)}^c$  when an class  $c$  object  $\beta^c \in O^c$  containing a variable  $x_{array(d)}^c$  is first instantiated. At this stage  $x_{array(d)}^c$  has not been constructed, and has no length. After a constructor call

$$x_{array(d)}^c = new_{array(d)}(e_{int}^c)$$

then  $x_{array(d)}^c$  now has the value  $(\perp_d, \dots, \perp_d)$  which is a vector of undefined values of length  $\llbracket e_{int}^c \rrbracket(\sigma, \beta^c)$  (assuming this is a positive value). Thus  $x_{array(d)}^c$  now has a well defined length and contains undefined values of type  $d$ .

(vi) The map

$$\llbracket \cdot \rrbracket : Exp(c)_d \times State \times O^c \rightarrow O^\perp^d$$

extends the definition in the notes by

$$\llbracket length(e_{array(d')^c}) \rrbracket(\sigma, \beta^c) = | \llbracket e_{array(d')^c} \rrbracket(\sigma, \beta^c) |$$

where  $|\cdot|$  is the usual length function on finite sequences.

If  $1 \leq \llbracket f_{int}^c \rrbracket(\sigma, \beta^c) \leq | \llbracket e_{array(d')^c} \rrbracket(\sigma, \beta^c) |$  then

$$\llbracket e_{array(d)}^c[f_{int}^c] \rrbracket(\sigma, \beta^c) = \llbracket e_{array(d)}^c \rrbracket(\sigma, \beta^c) \llbracket f_{int}^c \rrbracket(\sigma, \beta^c)$$

i.e. project out the array element numbered  $\llbracket f_{int}^c \rrbracket(\sigma, \beta^c)$  from the array, otherwise

$$\llbracket e_{array(d)}^c[f_{int}^c] \rrbracket(\sigma, \beta^c) = \perp_d.$$

(vii) A machine state will be a 4-tuple  $\langle S^c, \sigma, \beta^c, m \rangle$  where  $S^c$  is a currently executing statement belonging to class  $c \in C^{Array}$ , thus  $S^c \in Stat(c)$ , also  $\sigma \in State$  is the current memory state,  $\beta^c \in O^c$  is the currently executing object which has the thread of control and  $m \in \{nor, exc\}$  reflects the current status of the machine which can be normal (*nor*) or exception status (*exc*).

(viii) The transition rules for assignments are extended by the rules

$$\langle x_{array(d)}^c[e_{int}^c] = s_d^c, \sigma, \beta^c, nor \rangle \rightarrow \langle \epsilon, \sigma', \beta^c, nor \rangle$$

if  $1 \leq \llbracket e_{int}^c \rrbracket(\sigma, \beta^c) \leq | \llbracket x_{array(d)}^c \rrbracket(\sigma, \beta^c) |$  where

$$\sigma'_1 = \sigma_1, \sigma'_2 = \sigma_2, \sigma'_4 = \sigma_4, \sigma'_5 = \sigma_5,$$

and

$$\sigma'_3 = \sigma_3[ (\beta^c, x_{array(d)}^c) \llbracket e_{int}^c \rrbracket(\sigma, \beta^c) \mapsto \llbracket s_d^c \rrbracket(\sigma, \beta^c) ].$$

and for the exception case

$$\langle x_{array(d)}^c[e_{int}^c] = s_d^c, \sigma, \beta^c, nor \rangle \rightarrow \langle x_{array(d)}^c[e_{int}^c] = s_d^c, \sigma, \beta^c, exc \rangle$$

if  $\llbracket e_{int}^c \rrbracket(\sigma, \beta^c) < 1$  or  $\llbracket e_{int}^c \rrbracket(\sigma, \beta^c) > | \llbracket x_{array(d)}^c \rrbracket(\sigma, \beta^c) |$ .

(ix) The transition rules for constructors are extended by the rules

$$\langle x_{array(d)}^c = new_{array(d)}(e_{int}^c), \sigma, \beta^c, nor \rangle \rightarrow \langle \epsilon, \sigma', \beta^c, nor \rangle$$

if  $\llbracket e_{int}^c \rrbracket(\sigma, \beta^c) \geq 1$ , where

$$\sigma'_1 = \sigma_1, \sigma'_2 = \sigma_2, \sigma'_4 = \sigma_4, \sigma'_5 = \sigma_5,$$

and

$$\sigma'_3 = \sigma_3[ (\beta^c, x_{array(d)}^c) \mapsto (\perp_d, \dots, \perp_d) ].$$

and  $(\perp_d, \dots, \perp_d)$  is a list of undefined  $\perp_d$  elements exactly  $k$  long where  $k = \llbracket e_{int}^c \rrbracket(\sigma, \beta^c)$ .

For the exception case

$$\langle x_{array(d)}^c = new_{array(d)}(e_{int}^c), \sigma, \beta^c, nor \rangle \rightarrow$$

$$\langle x_{array(d)}^c = new_{array(d)}(e_{int}^c), \sigma, \beta^c, exc \rangle$$

if  $\llbracket e_{int}^c \rrbracket(\sigma, \beta^c) < 1$ .

(x) We assume that we start from a state  $\sigma$  in which every instance variable or attribute of a class  $c$  object  $\beta^c$  of type  $d \in C^{Array}$  has an undefined value  $\perp_d$ . Suppose  $\beta^c$  has the thread of control and no exceptions have been raised, i.e. the machine state is

$$\langle x_{array(int)}^c = new_{array(d)}(3); x_{array(int)}^c[2] = 0; x_{array(int)}^c[4] = 0; , \sigma, \beta^c, nor \rangle .$$

After executing the first instruction

$$x_{array(int)} := new_{array(int)}(3)$$

the abstract machine moves into the state

$$\langle x_{array(int)}^c[2] = 0; x_{array(int)}^c[4] = 0; , \sigma', \beta^c, nor \rangle .$$

We have a new state  $\sigma'$  with

$$\sigma'_1 = \sigma_1, \sigma'_2 = \sigma_2, \sigma'_4 = \sigma_4, \sigma'_5 = \sigma_5,$$

and

$$\sigma'_3 = \sigma_3[ (\beta^c, x_{array(int)}^c) \mapsto (\perp_{int}, \perp_{int}, \perp_{int}) ].$$

After executing the second instruction

$$x_{array(int)}[2] := 0;$$

the abstract machine moves into the state

$$\langle x_{array(int)}^c[4] = 0; , \sigma'', \beta^c, nor \rangle .$$

We have a new state  $\sigma''$  with

$$\sigma''_1 = \sigma'_1, \sigma''_2 = \sigma'_2, \sigma''_4 = \sigma'_4, \sigma''_5 = \sigma'_5,$$

and

$$\sigma''_3 = \sigma'_3[ (\beta^c, x_{array(int)}^c) \mapsto (\perp_{int}, 0, \perp_{int}) ].$$

After executing the third instruction

$$x_{array(int)}[4] := 0;$$

an exception is raised since the array access is out of bounds. The abstract machine moves into the state

$$\langle x_{array(int)}^c[4] = 0, \sigma'', \beta^c, exc \rangle$$

No more transitions are possible, and the abstract machine terminates execution in an exception state.