UNIVERSITY OF WESTMINSTER

SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

2009-2010

Code: 3SFE618 **Level:** 6

Title: Formal Methods

Date: 17 May 2010

Time: 10:00

Duration: 2 Hours

INSTRUCTIONS TO CANDIDATES

Answer ALL questions in Section A and TWO questions from Section B.

Section A is worth a total of 50 marks.

Each question in Section B is worth 25 marks.

You may wish to consult Appendix C.

Section A

Answer ALL questions from this section

Question 1

In relation to the process of developing a Z specification for an arbitrary system:

(a) What are the three categories of system states that need to be identified and differentiated? **[1 mark]**

(b) What is the *state schema* and what are its two main components? **[2 marks]**

(c) What are *state invariants* and what is their relationship to the three categories of system states? **[2 marks]**

(d) What is the purpose of the *initial state schema* and how does it relate to the *state schema*? **[2 marks]**

(e) When specifying a system operation it is usually necessary to define *pre-conditions* for the operation. What are *pre-conditions* and what is their purpose? **[3 marks]**

(f) The final stage in specifying a system operation is to define the *"total"* version of the operation. Explain what this means. **[2 marks]**

Question 2

Discuss how the following Z *schema calculus* operations enable large specifications to be written more easily.

- Schema Inclusion
- Schema Disjunction ($S \vee T$)
- Δ Convention (ΔS)
- Ξ Convention (ΞT)

[10 marks]

Question 3

The following is part of a specification for the *CarsRus* second hand car sales company. The following types representing individual cars, car manufacturers and car registration numbers.

$$
\begin{array}{ll}
CAR & ::= car1 \mid car2 \mid car3 \mid ... \\
MANUFACTURER & ::= Ford \mid BMW \mid Toyota \mid Honda \mid ... \\
REGNO & ::= R52\_ABC \mid S56\_PQR \mid T58\_XYZ \mid ...
\end{array}
$$

The state schema:

CarsRus

$stock$:
$manufacturers$:
$registration$:
$make$:
$price$:

$\vdots$

(a) Copy the *CarsRus* state schema and complete the definitions of the following state variables:

- $stock$ are the cars waiting to be sold.
- $manufacturers$ are the chosen cars manufacturers, e.g., Toyota, BMW, etc, whose cars *CarsRus* sells.
- $registration$ maps each car to its unique registration number.
- $make$ maps each car to its make.
- $price$ maps each car to price in pounds.

In addition, include the **state invariant** that *CarsRus* only sells cars that are made by its chosen manufacturers. **[12 marks]**

[Continued Overleaf]

(b) Define an **initial** $CarsRus$ state schema for the following situation:

- There are two cars $car1$ and $car2$.
- The chosen $manufacturers$ are Toyota, BMW and Honda.
- The registration numbers for $car1$ is $R52\_ABC$ and for $car2$ is $T58\_XYZ$.
- The make of $car1$ is BMW and $car2$ is a $Toyota$.
- $car1$ is £7000 and $car2$ is £5000.

[8 marks]

Question 4

Define the generic Z operator *domain restriction* "$\lhd$", using a generic schema. Its textual definition is given below.

Definition: The domain restriction, $S \lhd R$, of the relation R by the set S is the relation that relates x to y, if and only if, R relates x to y and x is a member of S.

[8 marks]

Section B

Answer TWO questions from this section

Question 5

The following is part of a Film DVD Rental Shop specification.

$$[\, FILM,\ DVD,\ CUSTOMER \,]$$

$$REPORT ::= OK$$

$$maxrentals : \mathbb{N}$$

FilmDataBase

$$stock : DVD \pfun FILM$$
$$customers : \mathbb{P}\, CUSTOMER$$

FilmRentals

$$rentedto : DVD \pfun CUSTOMER$$
$$inshop : \mathbb{P}\, DVD$$

$$\forall\, c : CUSTOMER \bullet \#(rentedto \rhd \{\ c\ \}) \leq maxrentals$$
$$inshop \cap \mathrm{dom}\ rentedto = \varnothing$$

FilmRentalShop

$$FilmDataBase$$
$$FilmRentals$$

$$\mathrm{dom}\ stock = inshop \cup \mathrm{dom}\ rentedto$$
$$\mathrm{ran}\ rentedto \subseteq customers$$

RentFilm_Success

$$
\begin{array}{l}
stock,\ stock' : DVD \pfun FILM \\
customers,\ customers' : \mathbb{P}\ CUSTOMER \\
rentedto,\ rentedto' : DVD \pfun CUSTOMER \\
inshop,\ inshop' : \mathbb{P}\ DVD \\
dvd? : DVD \\
customer? : CUSTOMER \\
report! : REPORT \\
\hline
\mathrm{dom}\ stock = inshop \cup \mathrm{dom}\ rentedto \\
\mathrm{dom}\ stock' = inshop' \cup \mathrm{dom}\ rentedto' \\
\mathrm{ran}\ rentedto \subseteq customers \\
\mathrm{ran}\ rentedto' \subseteq customers' \\
inshop \cap \mathrm{dom}\ rentedto = \varnothing \\
inshop' \cap \mathrm{dom}\ rentedto' = \varnothing \\
dvd? \in inshop \\
customer? \in customers \\
\#(rentedto \rhd \{\ customer?\ \}) < maxrentals \\
stock' = stock \\
customers' = customers \\
rentedto' = rentedto \oplus \{\ dvd? \mapsto customer?\ \} \\
inshop' = inshop \setminus \{\ dvd?\ \} \\
report! = OK
\end{array}
$$

(a) Explain in "plain English" (i.e., do not give a literal translation) the meaning of each of the following:

(i) The *invariants* of the *FilmRentalShop*. **[4 marks]**

(ii) The *pre-conditions* of the *RentFilm_Success* operation. **[3 marks]**

(b) Re-specify the *RentFilm_Success* operation so that it has a better structure and hence is more readable. This should be achieved by using the Schema Calculus and any new schemas that are required. **[8 marks]**

(c) Based on the version of *RentFilm_Success* specified as your answer to part **(b)**:

(i) Explain how the pre-conditions for the error cases of the operation are derived and then derive them. **[3 marks]**

(ii) Define a *"total" RentFilm* operation using the schema calculus and any necessary additional types and schemas. **[7 marks]**

Question 6

Figure 1 represents a virtual terminal window, the four areas correspond to how the window is to be divided up to represent the four display panels (Navigation, Title, Goods and Shopping Cart) for a web page for an online shopping web site. The window is 80 columns wide and 24 rows high.

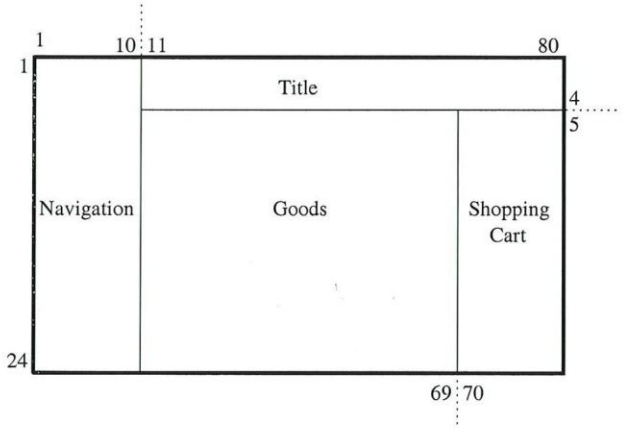

Figure 1: Shopping web page's display panels

(a) Define suitable Z types and constants to represent the virtual window as a whole. **[7 marks]**

(b) Define suitable Z constants to represent the four display panel areas within the virtual window, i.e., Navigation, Title, Goods and Shopping Cart. **[12 marks]**

(c) As part of the development process a *MouseLocationTest* operation is required. This operation simply inputs the location of the mouse within the virtual window and outputs the identity of the panel that the mouse is in. Note that this operation does not have any state, but simply uses the display panel areas to determine the location of the mouse. **[6 marks]**

Question 7

The partial Z specification of a VDU screen which allows cursor key movements is given in Appendix A.

(a) Discuss how the two Z tools ZTC and ZANS assist in the development of a Z specification? **[4 marks]**

(b) The ZTC type checker output for the cursor key specification is given in Appendix B. For each error give an explanation and the necessary corrections. **[11 marks]**

(c) Once all of the errors detailed in part **(b)** have been eliminated from the cursor keys specification, explain what additions and modifications must be made to the specification to permit it to be animated by ZANS. **[10 marks]**

Appendix A. Cursor Keys Specification

A.1 ZTC Box Style Version

```
1           specification
2
3           XY_COORDINATE == N & N
4
5           | numbcols, numbrows : N1
6
7           KEY ::=   DownKEY | LeftKey | CurrentPositionKey
8
9           ERROR ::=  OnLastRow_CanNotMove_Down
10                   | AtHome_CanNotMove_Left
11
12          REPORT ::= Success | error << ERROR >>
13
14          | SCREEN : N <-> N
15
16          | HomePosition : XY_COORDINATE ;
17
18          --- Cursor ---------------------------------------
19          | position : XY_COORDINATE
20          |------------------------------
21          | position subseteq SCREEN
22          ------------------------------------------------
23
24          --- InitialCursor ------------------------------
25          | Cursor
26          |-----------------------------
27          | position = HomePosition
28          ------------------------------------------------
29
30          ReportSuccess =^= [ report! : REPORT | report! = Success ]
31
32          Pressed_DownKey =^= [ key? : KEY | key? = DownKey ]
33
34          --- Down_NotLastRow ----------------------------
35          | Delta Cursor
36          |--------------------
37          | second(position) < numbrows
```

```
38          | position' = ( first(position), second(position) + 1 )
39          ------------------------------------------------
40
41          --- Down_OnLastRow_CanNotMove -------------------
42          | Xi Cursor ;
43          | report! : REPORT
44          |--------------------------
45          | second(position) = numbrows ;
46          | report! = OnLastRow_CanNotMove_Down
47          ------------------------------------------------
48
49          Down_Success =^= Down_NotLastRow /\ ReportSuccess
50
51          Down_Errors =^= Down_OnLastRow_CanNotMove
52
53          Down =^= Pressed_DownKey /\ ( Down_Success \/ Down_ERRORS )
54
55          Pressed_LeftKey =^= [ key? : KEY | key? = LeftKey ]
56
57          --- Left_NotFirstColumn -------------------------
58          | Delta Cursor
59          |--------------------------
60          | first(position) > 1 ;
61          | position' = { first(position) - 1, second(position) }
62          ------------------------------------------------
63
64          --- Left_FirstColumnNotFirstRow -----------------
65          | Cursor ;
66          |--------------------------
67          | first(position) = 1 ;
68          | second(position) > 1 ;
69          | position' = ( numbcols, second(position) - 1 )
70          ------------------------------------------------
71
72          --- Left_AtHome_CanNotMove ----------------------
73          | Xi Cursor ;
74          | report : REPORT
75          |--------------------------
76          | position = HomePosition ;
77          | report! = error(AtHome_CanNotMove_Left)
78          ------------------------------------------------
```

```
79
80          Left_Success =^= (     Left_NotFirstColumn
81                             \/  Left_FirstColumnNotFirstRow )
82                           /\ ReportSuccess
83
84          Left_Errors =^= Left_AtHome_CanNotMove
85
86
87          Left =^=     Pressed_LeftKey
88                    /\ ( Left_Success \/ Left_Errors )
```

A.2 LaTeX 2_ε Version

$$XY\_COORDINATE == \mathbb{N} \times \mathbb{N}$$

$$| \; numbcols, numbrows : \mathbb{N}_1$$

$$KEY ::= DownKEY \mid LeftKey \mid CurrentPositionKey$$

$$ERROR ::= OnLastRow\_CanNotMove\_Down \mid AtHome\_CanNotMove\_Left$$

$$REPORT ::= Success \mid error\langle\langle ERROR \rangle\rangle$$

$$| \; SCREEN : \mathbb{N} \leftrightarrow \mathbb{N}$$

$$| \; HomePosition : XY\_COORDINATE$$

Cursor

$$position : XY\_COORDINATE$$

$$position \subseteq SCREEN$$

InitialCursor

$$Cursor$$

$$position = HomePosition$$

$$ReportSuccess \mathrel{\widehat=} [report! : REPORT \mid report! = Success]$$
$$Pressed\_DownKey \mathrel{\widehat=} [key? : KEY \mid key? = DownKey]$$

Down_NotLastRow

$$\Delta Cursor$$

$$second(position) < numbrows$$
$$position' = (first(position), second(position) + 1)$$

Down_OnLastRow_CanNotMove

$$\Xi Cursor$$
$$report! : REPORT$$

$$second(position) = numbrows$$
$$report! = OnLastRow\_CanNotMove\_Down$$

$$Down\_Success \mathrel{\widehat=} Down\_NotLastRow \land ReportSuccess$$
$$Down\_Errors \mathrel{\widehat=} Down\_OnLastRow\_CanNotMove$$
$$Down \mathrel{\widehat=} Pressed\_DownKey \land (Down\_Success \lor Down\_ERRORS)$$

$$Pressed\_LeftKey \mathrel{\widehat=} [key? : KEY \mid key? = LeftKey]$$

Left_NotFirstColumn

$$\Delta Cursor$$

$$first(position) > 1$$
$$position' = \{first(position) - 1, second(position)\}$$

Left_FirstColumnNotFirstRow

$$Cursor$$

$$first(position) = 1$$
$$second(position) > 1$$
$$position' = (numbcols, second(position) - 1)$$

$$
\begin{array}{|l}
\hline
\textit{Left\_AtHome\_CanNotMove} \\
\Xi Cursor \\
report : REPORT \\
\hline
position = HomePosition \\
report! = error(AtHome\_CanNotMove\_Left) \\
\hline
\end{array}
$$

$$Left\_Success \mathrel{\widehat=} (Left\_NotFirstColumn \lor Left\_FirstColumnNotFirstRow) \land ReportSuccess$$

$$Left\_Errors \mathrel{\widehat=} Left\_AtHome\_CanNotMove$$

$$Left \mathrel{\widehat=} Pressed\_LeftKey \land (Left\_Success \lor Left\_Errors)$$

Appendix B. ZTC Type Check Log File

```
... Initializing.
... Loading Z mathematical tools library: math0.zbx
Parsing main file: cursor.zbx
... Type checking Equivalence definition: XY_COORDINATE. "cursor.zbx" Line 3
... Type checking Axiom box. "cursor.zbx" Line 5
... Type checking Free type definition: KEY. "cursor.zbx" Line 7
... Type checking Free type definition: ERROR. "cursor.zbx" Lines 9-10
... Type checking Free type definition: REPORT. "cursor.zbx" Line 12
... Type checking Axiom box. "cursor.zbx" Line 14
... Type checking Axiom box. "cursor.zbx" Line 16
... Type checking Schema box: Cursor. "cursor.zbx" Lines 18-21
--- Typing error. "cursor.zbx" Line 21. Type mismatch: Infix relation.
... Type checking Schema box: InitialCursor. "cursor.zbx" Lines 24-27
... Type checking Schema definition: ReportSuccess. "cursor.zbx" Line 30
... Type checking Schema definition: Pressed_DownKey. "cursor.zbx" Line 32
--- Typing error. "cursor.zbx" Line 32. Undefined name: DownKey
... Type checking Schema box: Down_NotLastRow. "cursor.zbx" Lines 34-38
--- Typing error. "cursor.zbx" Line 37. Mapping expected:
... Type checking Schema box: Down_OnLastRow_CanNotMove. "cursor.zbx" Lines 41-46
--- Typing error. "cursor.zbx" Line 46. Type mismatch:
... Type checking Schema definition: Down_Success. "cursor.zbx" Line 49
... Type checking Schema definition: Down_Errors. "cursor.zbx" Line 51
--- Syntax error. "cursor.zbx" Line 53, near "Down_ERRORS"
... Type checking Schema definition: Pressed_LeftKey. "cursor.zbx" Line 55
... Type checking Schema box: Left_NotFirstColumn. "cursor.zbx" Lines 57-61
--- Typing error. "cursor.zbx" Line 61. Type mismatch:
... Type checking Schema box: Left_FirstColumnNotFirstRow. "cursor.zbx" Lines 64-69
--- Typing error. "cursor.zbx" Line 69. Undefined name: position'
... Type checking Schema box: Left_AtHome_CanNotMove. "cursor.zbx" Lines 72-77
--- Typing error. "cursor.zbx" Line 77. Undefined name: report!
... Type checking Schema definition: Left_Success. "cursor.zbx" Lines 80-82
... Type checking Schema definition: Left_Errors. "cursor.zbx" Line 84
--- Reached the end of the main file while parsing.
... Type checking Schema definition: Left. "cursor.zbx" Lines 87-88
End of main file: cursor.zbx
Log written in "cursor.log"
```

Appendix C. Table of Z Syntax

This appendix contains the Z notation for: sets, logic, ordered pairs, relations, functions, sequences, bags and schemas.

C.1 Sets

Notation	Description
$\mathbb{N}$	Set of natural numbers from 0
$\mathbb{N}_1$	Set of natural numbers from 1
$\mathbb{Z}$	Set of integers
$x \in S$	x is an element of S
$x \notin S$	x is not an element of S
$S \subseteq T$	S is a subset of T
$S \subset T$	S is a strict subset of T
$\varnothing$, $\{\ \}$	Empty set
$\mathbb{P}\, S$	Power set of S
$\mathbb{F}\, S$	Finite power set of S
$\mathbb{F}_1\, S$	Non-empty finite subsets of S
$S \cup T$	Union of S and T
$S \cap T$	Intersection of S and T
$S \setminus T$	Set difference of S and T
$\#S$	Number of elements in set S
$\{\ D \mid P \bullet t\ \}$	Set comprehension
$\bigcup SS$	Distributed union of SS
$\bigcap SS$	Distributed intersection of SS
$i\,..\,j$	Range of integers from i to j inclusive
disjoint $\langle A, B, C\rangle$	Disjoint sets A, B and C
$\langle A, B, C\rangle$ partition S	Sets A, B and C partition the set S

C.2 Logic

Notation	Description
$\neg\, P$	not P
$P \wedge Q$	P and Q
$P \vee Q$	P or Q
$P \Rightarrow Q$	P implies Q
$P \Leftrightarrow Q$	P is equivalent to Q
$\forall x : T \bullet P$	All elements x of type T satisfy P
$\exists x : T \bullet P$	There exists an element x of type T which satisfies P

C.3 Ordered Pairs

Notation	Description
$X \times Y$	Cartesian product of X and Y
(x, y)	Ordered pair
$x \mapsto y$	Ordered pair, (maplet)
$\mathit{first}(x, y)$	Ordered pair projection function
$\mathit{second}(x, y)$	Ordered pair projection function

C.4 Relations

Notation	Description
$\mathbb{P}(X \times Y)$	Set of relations between X and Y
$X \leftrightarrow Y$	Set of relations between X and Y
$\mathrm{dom}\, R$	Domain of relation R
$\mathrm{ran}\, R$	Range of relation R
$S \lhd R$	Domain restriction of R to the set S
$S \ntriangleleft R$	Domain anti-restriction of R by the set S
$R \rhd S$	Range restriction of R to the set S
$R \ntriangleright S$	Range anti-restriction of R by the set S
$R_1 \oplus R_2$	R_1 overridden by relation R_2
$R \fatsemi Q$	Relational composition
$R(\!\| S \|\!)$	Relational Image of the set S of relation R
$\mathrm{id}\, X$	Identity relation
R^{-1}	Inverse relation
R^{+}	Transitive closure of R
R^{*}	Reflexive-transitive closure of R

C.5 Functions

Notation	Description
$\twoheadrightarrow\!\!\!\!\!\!\!\shortmid\shortmid$ (⇸⇸)	Finite function
⤔⇸	Finite injection
$\nrightarrow$	Partial function
$\longrightarrow$	Total function
$\rightarrowtail\!\!\!\!\!\shortmid$ (⤔)	Partial injection
$\rightarrowtail$	Total injection
⤀	Partial surjection
$\twoheadrightarrow$	Total surjection
⤖	Bijection

C.6 Sequences

Notation	Description
$\mathrm{seq}\, X$	Finite sequences of type X
$\mathrm{seq}_1\, X$	Non-empty finite sequences of type X
iseq X	Injective finite sequences of type X
$\langle\, \rangle$	Empty sequence
$s \frown t$	Concatenation of the sequences s and t
$head\ s$	First element of a non empty sequence
$tail\ s$	All but first element of a non empty sequence
$last\ s$	Last element of a non empty sequence
$front\ s$	All but last element of a non empty sequence
$rev\ s$	Sequence Reversal
$squash\ s$	Sequence Compaction
s prefix t	s is a *prefix* of t
s suffix t	s is a *suffix* of t
$s \,\mathrm{in}\, t$	s is a *sub-sequence* of t

C.7 Bags

Notation	Description
$\mathrm{bag}\, X$	Bag of type X
$[\![ ]\!]$	Empty bag
$x \Subset B$	Bag membership
$x \notin B$	Bag non-membership
$B_1 \sqsubseteq B_2$	Sub-bag
$B_1 \sqsubset B_2$	Strict Sub-bag
$B_1 \uplus B_2$	Bag Union
$B_1 \setminus\!\!\setminus B_2$	Bag Difference
$count\ B\ x$	Bag Count of x
$B \sharp x$	Bag Count of x
$n \otimes B$	Bag Scaling
$items\ s$	Bag of the sequence s

C.8 Schemas

Schema Type	Schema Box
Axiom	*declarations* ——— *constraints*
Generic	$[X, \ldots]$ *declarations* ——— *constraints*
Linear	$S \mathrel{\widehat=} [declarations \mid constraints]$
State/Operation	S *declarations* ——— *constraints*

C.9 Schema Calculus

Notation	Description
$[S;\ declarations \mid constraints]$	Schema inclusion
S'	Schema decoration
ΔS	Δ (Delta) Convention
ΞS	Ξ (Xi) Convention
$S \wedge T$	Schema Conjunction (S and T)
$S \vee T$	Schema Disjunction (S or T)