UNIVERSITY OF WESTMINSTER

ELECTRONICS AND COMPUTER SCIENCE

2012-2013

Code:	ECSE610
Title:	Formal Specification
Date:	14 May 2013
Time:	10:00
Duration:	2 Hours
Module Leader:	Paul Howells

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 the Z notation given in Appendix C.

Section A

Answer ALL questions from this section.
You may wish to consult the Z notation given in Appendix C.

Question 1

Given the following axiom schema and two state schemas:

$$\begin{array}{|l} MaxLength : \mathbb{N} \\ \hline MaxLength = 10 \end{array}$$

$$\begin{array}{|l} \text{— } Sets \text{ —} \\ A, B : \mathbb{P}\,\mathbb{N} \\ x, y : \mathbb{N} \\ \hline x > y \\ x \in A \\ y \notin B \\ \hline \end{array}$$

$$\begin{array}{|l} \text{— } Sequences \text{ —} \\ list1, list2 : \operatorname{seq} \mathbb{N} \\ z : \mathbb{N} \\ \hline list1 \text{ prefix } list2 \\ \# list2 < MaxLength \\ \langle 0, 1, 2, z \rangle \text{ in } list2 \\ \hline \end{array}$$

Give the expanded versions of the following two schemas:

(a) $\Delta Sets$ **[4 marks]**

(b) $\Xi Sequences$ **[6 marks]**

Question 2

Given the following Z declarations:

$$SHAPE ::= Oval \mid Circle \mid Triangle \mid Rectangle \mid Square \mid Rhombus \mid Pentagon \mid Hexagon$$

$$\begin{array}{|l}
Quadrilaterals : \mathbb{P}\, SHAPE \\
NonPolygons : \mathbb{P}\, SHAPE \\
\hline
Quadrilaterals = \{\ Rectangle, Square, Rhombus\ \} \\
NonPolygons = \{\ Oval, Circle\ \}
\end{array}$$

$$\begin{array}{|l}
edges : SHAPE \rightarrow \mathbb{N} \\
\hline
edges = \{\ (Oval, 1),\ (Circle, 1),\ (Triangle, 3),\ (Rectangle, 4), \\
\qquad (Square, 4),\ (Rhombus, 4),\ (Pentagon, 5),\ (Hexagon, 6)\ \}
\end{array}$$

Evaluate the following expressions:

(a) $Quadrilaterals \cup NonPolygons$ **[1 mark]**

(b) $Quadrilaterals \setminus \{Rhombus\}$ **[1 mark]**

(c) $\#edges$ **[1 mark]**

(d) $\mathrm{dom}\ edges$ **[1 mark]**

(e) $\mathrm{ran}\ edges$ **[1 mark]**

(f) $edges(Pentagon)$ **[1 mark]**

(g) $\mathbb{P}\, NonPolygons$ **[2 marks]**

(h) $edges \rhd \{4\}$ **[2 marks]**

(i) $NonPolygons \lhd edges$ **[2 marks]**

(j) $(\ Quadrilaterals \cup NonPolygons\ ) \mathbin{⩤} edges$ **[3 marks]**

Question 3

Given the following definitions of the two relations R and Q:

$$
\begin{array}{|l}
R : \mathbb{N} \leftrightarrow \mathbb{N} \\
Q : \mathbb{N} \leftrightarrow \mathbb{N} \\
\hline
R = \{\, 0 \mapsto 0,\ 1 \mapsto 2,\ 2 \mapsto 3,\ 3 \mapsto 3,\ 3 \mapsto 4,\ 3 \mapsto 5,\ 4 \mapsto 5 \,\} \\
Q = \{\, 0 \mapsto 1,\ 3 \mapsto 3,\ 4 \mapsto 5,\ 4 \mapsto 6,\ 5 \mapsto 5,\ 6 \mapsto 7 \,\}
\end{array}
$$

(a) Evaluate the following expressions:

(i) $Q \llparenthesis \{3,4\} \rrparenthesis$ **[2 marks]**

(ii) $Q \oplus \{\, (3,7) \,\}$ **[2 marks]**

(iii) $R \oplus \{\, (3,7) \,\}$ **[3 marks]**

(iv) $R \,⨾\, Q$ **[5 marks]**

(b) Explain why R and Q are relations but not functions. **[3 marks]**

Question 4

The following are examples of the standard mathematical functions for *increment*, *decrement*, *addition* and *subtraction* respectively, for natural numbers ($\mathbb{N}$).

$$
\begin{aligned}
inc(7) &= 8 \\
dec(1) &= 0 \\
add(9,3) &= 12 \\
sub(11,5) &= 6
\end{aligned}
$$

Define the *signatures* of these functions. Note that you are **not** required to give their definitions.

(a) inc and dec **[4 marks]**

(b) add and $minus$ **[6 marks]**

Section B

Answer TWO questions from this section.
You may wish to consult the Z notation given in Appendix C.

Question 5

The following is part of a specification for the *MobilesRUs* mobile phone shop. The following types are used to represent individual phones, mobile manufacturers and unique mobile serial numbers.

$$
\begin{array}{ll}
MOBILE & ::= mob1 \mid mob2 \mid mob3 \mid mob4 \mid mob5 \mid \ldots \\
MANUFACTURER & ::= Samsung \mid HTC \mid Apple \mid Nokia \mid Blackberry \mid \ldots \\
SERIAL\_NO & ::= MP111 \mid MP012 \mid MP212 \mid MP008 \mid MP099 \mid \ldots
\end{array}
$$

The current state of the $MobilesRUs$ shop is represented by the state schema:

$MobilesRUs$
$stock$:
$manufacturers$:
$serialNumber$:
$make$:
$price$:
⋮

(a) Copy and complete the definition of the $MobilesRUs$ state schema by:

(i) Completing the definition of the following state variables:

- $stock$ are the mobile phones waiting to be sold.
- $manufacturers$ are the chosen phone manufacturers, e.g., Nokia, Samsung, HTC, etc, whose phones *MobilesRUs* sells.
- $serialNumber$ maps each mobile to its *unique* serial number.
- $make$ maps each mobile to its make, i.e. manufacturer.
- $price$ maps each phone to its price in (whole) pounds. **[9 marks]**

(ii) Including the **state invariant** that *MobilesRUs* only sells phones that are made by its chosen manufacturers. **[3 marks]**

[Continued Overleaf]

(b) Define an **initial state schema** for the first day of opening for the *MobilesRUs* shop, when it has just two phones to sell:

- The two phones are $mob1$ and $mob2$.
- The chosen $manufacturers$ are Samsung, Nokia, HTC and Blackberry.
- The registration numbers for $mob1$ is $MP111$ and for $mob2$ is $MP212$.
- The make of $mob1$ is HTC and $mob2$ is $Samsung$.
- The cost of the mobiles are as follows: $mob1$ is worth £350 and $mob2$ is £225.

[8 marks]

(c) The $MobilesRUs$ shop manager realises that he needs to keep track of what operating system (OS) each phone runs, as lots of customers want to make sure they have a phone with a particular OS. For example, not a Windows phone, but an Android one.

Define a suitable Z type to represent the types of phone OS and give the signature for a new state variable called $phoneOS$ that could be added to the state schema to record this information for each phone. **[5 marks]**

Question 6

The following is part of a Car Hire Company specification.
The following definitions represent the models of car they hire out, the actual cars they have to hire out and companies that hire them.

$$[\ MODEL,\ CAR,\ COMPANY\ ]$$

$$HireLimit : \mathbb{N}$$

CarDataBase

$$hireCars : CAR \twoheadrightarrow\!\!\!\!\!\!\!\!\;\; MODEL$$

$$registeredCompanies : \mathbb{F}\ COMPANY$$

Hirings

$$hiredto : CAR \twoheadrightarrow\!\!\!\!\!\!\!\!\;\; COMPANY$$

$$inGarage : \mathbb{F}\ CAR$$

$$\forall\, co : COMPANY \bullet \#(hiredto \rhd \{\ co\ \}) \leq HireLimit$$

$$inGarage \cap \operatorname{dom} hiredto = \varnothing$$

CarHireCompany

$$CarDataBase$$

$$Hirings$$

$$\operatorname{dom} hireCars = inGarage \cup \operatorname{dom} hiredto$$

$$\operatorname{ran} hiredto \subseteq registeredCompanies$$

[Continued Overleaf]

HireACar

$$\begin{array}{l}
\Delta CarHireCompany \\
car? : CAR \\
company? : COMPANY \\
\hline
car? \in inGarage \\
hireCars' = hireCars \\
inGarage' = inGarage \setminus \{ car? \} \\
\#(hiredto \rhd \{ company? \}) < HireLimit \\
registeredCompanies' = registeredCompanies \\
company? \in registeredCompanies \\
hiredto' = hiredto \oplus \{ car? \mapsto company? \}
\end{array}$$

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

(i) *Hirings* **[3 marks]**

(ii) *CarHireCompany* **[3 marks]**

(b) Explain in "plain English" the meaning of each line of the **constraint** part of the *HireACar* schema and the role it plays in the specification of the operation. **[7 marks]**

(c) Specify the *ReturnHireCar* operation which is used when a Company returns a hire car to the Car Hire Company. The specification of this operation must be total and output appropriate success and error reports. In addition the specification should be as modular as possible and make full use of the schema calculus.

[Marks Guide: types 1 mark, successful case 6 marks, error cases 4 marks, total operation 1 mark.] **[12 marks]**

Question 7

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

(a) Discuss how the two Z tools ZTC and ZANS assist in the development of a Z specification? **[7 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. **[10 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. **[8 marks]**

Appendix A. Cursor Keys Specification

A.1 ZTC Box Style Version

```
specification

XY_COORDINATE == N & N

| numbcols, numbrows : N1

KEY ::=   DownKEY | LeftKey | CurrentPositionKey

REPORT ::= Success
         | ERROR_OnLastRow_CanNotMove_Down
         | ERROR_AtHome_CanNotMove_Left

| SCREEN : N <-> N

| HomePosition : XY_COORDINATE ;

--- Cursor -------------------------------------
| position : XY_COORDINATE
|------------------------------
| position subseteq SCREEN
------------------------------------------------

--- InitialCursor ------------------------------
| Cursor
|------------------------------
| position = HomePosition
------------------------------------------------

ReportSuccess =^= [ report! : REPORT | report! = Success ]

Pressed_DownKey =^= [ key? : KEY | 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! = ERROR_OnLastRow_CanNotMove_Down
--------------------------------------------------

Down_Success =^=  Down_NotLastRow /\ ReportSuccess

Down_Errors =^= Down_OnLastRow_CanNotMove

Down =^= Pressed_DownKey /\ ( Down_Success \/ Down_ERRORS )

Pressed_LeftKey =^= [ key? : KEY | 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 )
--------------------------------------------------

--- Left_AtHome_CanNotMove ----------------------
| Xi Cursor ;
| report! : REPORT
|--------------------------
| position = HomePosition ;
| report! = ERROR_AtHome_CanNotMove_Left
--------------------------------------------------
```

```
        Left_Success =^= (      Left_NotFirstColumn
                              \/  Left_FirstColumnNotFirstRow )
                            /\ ReportSuccess

        Left_Errors =^= Left_AtHome_CanNotMove


        Left =^=      Pressed_LeftKey
                  /\ ( Left_Success \/ Left_Errors )
```

A.2 LaTeX 2_ε Version

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

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

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

$$\begin{aligned} REPORT ::= {} & Success \\ \mid {} & ERROR\_OnLastRow\_CanNotMove\_Down \\ \mid {} & ERROR\_AtHome\_CanNotMove\_Left \end{aligned}$$

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

$$\mid \; HomePosition : XY\_COORDINATE$$

$$\begin{array}{|l} \hline \textit{Cursor} \\ position : XY\_COORDINATE \\ \hline position \subseteq SCREEN \\ \hline \end{array}$$

$$\begin{array}{|l} \hline \textit{InitialCursor} \\ Cursor \\ \hline position = HomePosition \\ \hline \end{array}$$

$ReportSuccess \mathrel{\widehat=} [report! : REPORT \mid report! = Success]$

$Pressed\_DownKey \mathrel{\widehat=} [key? : KEY \mid key? = DownKey]$

$$\begin{array}{|l}
\textit{Down_NotLastRow} \\ \hline
\Delta Cursor \\ \hline
second(position) < numbrows \\
position' = (first(position), second(position) + 1) \\ \hline
\end{array}$$

$$\begin{array}{|l}
\textit{Down_OnLastRow_CanNotMove} \\ \hline
\Xi Cursor \\
report : REPORT \\ \hline
second(position) = numbrows \\
report! = ERROR_OnLastRow_CanNotMove_Down \\ \hline
\end{array}$$

$Down\_Success \mathrel{\widehat=} Down\_NotLastRow \land ReportSuccess$

$Down\_Errors \mathrel{\widehat=} Down\_OnLastRow\_CanNotMove$

$Down \mathrel{\widehat=} Pressed\_DownKey \land (Down\_Success \lor Down\_ERRORS)$

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

$$\begin{array}{|l}
\textit{Left_NotFirstColumn} \\ \hline
\Delta Cursor \\ \hline
first(position) > 1 \\
position' = \{first(position) - 1, second(position)\} \\ \hline
\end{array}$$

$$\begin{array}{|l}
\textit{Left_FirstColumnNotFirstRow} \\ \hline
Cursor \\ \hline
first(position) = 1 \\
second(position) > 1 \\
position' = (numbcols, second(position) - 1) \\ \hline
\end{array}$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\_\_ \mathit{Left\_AtHome\_CanNotMove} \_\_\_\_} \\
\Xi \mathit{Cursor} \\
\mathit{report!} : \mathit{REPORT} \\
\hline
\mathit{position} = \mathit{HomePosition} \\
\mathit{report!} = \mathit{ERROR\_AtHome\_CanNotMove\_Left} \\
\hline
\end{array}
$$

$$
\begin{aligned}
\mathit{Left\_Success} &\mathrel{\widehat=} (\mathit{Left\_NotFirstColumn} \lor \mathit{Left\_FirstColumnNotFirstRow}) \\
&\qquad \land \mathit{ReportSuccess} \\
\mathit{Left\_Errors} &\mathrel{\widehat=} \mathit{Left\_AtHome\_CanNotMove} \\
\mathit{Left} &\mathrel{\widehat=} \mathit{Pressed\_LeftKey} \land (\mathit{Left\_Success} \lor \mathit{Left\_Errors})
\end{aligned}
$$

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, schemas and the schema calculus.

C.1 Sets

<table>
<tr><th>Z Notation</th><th>ZTC</th><th>Description</th></tr>
<tr><td>$\mathbb{N}$</td><td>N</td><td>Set of natural numbers from 0</td></tr>
<tr><td>$\mathbb{N}_1$</td><td>N1</td><td>Set of natural numbers from 1</td></tr>
<tr><td>$\mathbb{Z}$</td><td>Z</td><td>Set of integers</td></tr>
<tr><td>$x \in S$</td><td>x in S</td><td>x is an element of S</td></tr>
<tr><td>$x \notin S$</td><td>x notin S</td><td>x is not an element of S</td></tr>
<tr><td>$S \subseteq T$</td><td>S subset T</td><td>S is a subset of T</td></tr>
<tr><td>$S \subset T$</td><td>S subseteq T</td><td>S is a strict subset of T</td></tr>
<tr><td>$\varnothing$, $\{\,\}$</td><td>{}</td><td>Empty set</td></tr>
<tr><td>$\mathbb{P} S$</td><td>P S</td><td>Power set of S</td></tr>
<tr><td>$\mathbb{F} S$</td><td>F S</td><td>Finite power set of S</td></tr>
<tr><td>$S \cup T$</td><td>S || T</td><td>Union of S and T</td></tr>
<tr><td>$S \cap T$</td><td>S && T</td><td>Intersection of S and T</td></tr>
<tr><td>$S \setminus T$</td><td>S \ T</td><td>Set difference of S and T</td></tr>
<tr><td>$\#S$</td><td>#S</td><td>Number of elements in set S</td></tr>
<tr><td>$\{ D \mid P \bullet E \}$</td><td>{ D | P @ E }</td><td>Set comprehension</td></tr>
<tr><td>$\bigcup SS$</td><td>Union SS</td><td>Distributed union of SS</td></tr>
<tr><td>$\bigcap SS$</td><td>Intersection SS</td><td>Distributed intersection of SS</td></tr>
<tr><td>$i \mathrel{..} j$</td><td>i..j</td><td>Range of integers from i to j inclusive</td></tr>
<tr><td>disjoint $\langle A, B, C \rangle$</td><td>disjoint << A, B, C >></td><td>Disjoint sets A, B and C</td></tr>
<tr><td>$\langle A, B, C \rangle$ partition S</td><td><< A, B, C >> partition S</td><td>Sets A, B and C partition the set S</td></tr>
</table>

C.2 Logic

Z Notation	ZTC	Description
$\neg P$	`not P`	not P
$P \land Q$	`P and Q`	P and Q
$P \lor Q$	`P or Q`	P or Q
$P \Rightarrow Q$	`P => Q`	P implies Q
$P \Leftrightarrow Q$	`P <=> Q`	P is equivalent to Q
$\forall x : T \bullet P$	`forall x : T @ P`	All elements x of type T satisfy P
$\exists x : T \bullet P$	`exists x : T @ P`	There exists an element x of type T which satisfies P
$\exists_1 x : T \bullet P$	`exists1 x : T @ P`	There exists a *unique* element x of type T which satisfies P

C.3 Ordered Pairs

Notation	ZTC	Description
$X \times Y$	`X & Y`	Cartesian product of X and Y
(x, y)	`(x, y)`	Ordered pair
$x \mapsto y$	`x -> y`	Ordered pair, (maplet)
$first(x, y)$	`first(x, y)`	Ordered pair projection function
$second(x, y)$	`second(x, y)`	Ordered pair projection function

C.4 Relations

Notation	ZTC	Description
$\mathbb{P}(X \times Y)$	`P(X & Y)`	Set of relations between X and Y
$X \leftrightarrow Y$	`X <-> Y`	Set of relations between X and Y
dom R	`dom R`	Domain of relation R
ran R	`ran R`	Range of relation R
$S \lhd R$	`S <| R`	Domain restriction of R to the set S
$S \mathbin{⩤} R$	`S <+ R`	Domain anti-restriction of R by the set S
$R \rhd S$	`R |> S`	Range restriction of R to the set S
$R \mathbin{⩥} S$	`R +> S`	Range anti-restriction of R by the set S
$R_1 \oplus R_2$	`R1 += R2`	R_1 overridden by relation R_2
$R \mathbin{⨾} Q$	`R :> Q`	Relational composition
$R(\!\mid S \mid\!)$	`R (| S |)`	Relational Image of the set S of relation R
id X	`id X`	Identity relation
R^{-1}	`R~`	Inverse relation
R^+	`R^+`	Transitive closure of R
R^*	`R^*`	Reflexive-transitive closure of R

C.5 Functions

Notation	ZTC	Description
⇻	`++>`	Finite function
⤕	`>++>`	Finite injection
⇸	`+->`	Partial function
$\rightarrow$	`-->`	Total function
⤔	`>+>`	Partial injection
$\rightarrowtail$	`>->`	Total injection
⤀	`+>>`	Partial surjection
$\twoheadrightarrow$	`->>`	Total surjection
⤖	`>->>`	Bijection

C.6 Sequences

Notation	ZTC	Description
$\mathrm{seq}\, X$	`seq X`	Finite sequences of type X
$\mathrm{seq}_1\, X$	`seq1 X`	Non-empty finite sequences of type X
$\mathrm{iseq}\, X$	`iseq X`	Injective finite sequences of type X
$\langle\, \rangle$	`<<>>`	Empty sequence
$s \frown t$	`s ^ t`	Concatenation of the sequences s and t
$\mathit{head}\ s$	`head s`	First element of a non empty sequence
$\mathit{tail}\ s$	`tail s`	All but first element of a non empty sequence
$\mathit{last}\ s$	`last s`	Last element of a non empty sequence
$\mathit{front}\ s$	`front s`	All but last element of a non empty sequence
$\mathit{rev}\ s$	`rev s`	Sequence Reversal
$\mathit{squash}\ s$	`squash s`	Sequence Compaction
s prefix t	`s prefix t`	s is a *prefix* of t
s suffix t	`s suffix t`	s is a *suffix* of t
$s \mathrm{in}\, t$	`s subseq t`	s is a *sub-sequence* of t

C.7 Schema Calculus

Z Notation	ZTC	Description
$[S;\ D \mid C]$	`[ S; D | C ]`	Schema inclusion
S'	`S'`	Schema decoration
ΔS	`Delta S`	Δ (Delta) Convention
ΞS	`Xi S`	Ξ (Xi) Convention
$S \wedge T$	`S and T`	Schema Conjunction (S and T)
$S \vee T$	`S or T`	Schema Disjunction (S or T)

C.8 Schemas Types: Z & ZTC Schema Boxes

Axiom Schema

| *declarations*
|---
| *constraints*

```
| declarations
|--------------
| constraints
```

Generic

$[X, \ldots]$
declarations
---
constraints

```
===[X, Y]=================
| declarations
|--------------
| constraints
--------------------------
```

Linear Schema

$S \mathrel{\widehat=} [\mathit{declarations} \mid \mathit{constraints}]$

```
S =^= [ declarations | constraints ]
```

State/Operation Schema

Operation
declarations
---
constraints

```
--- Operation ------------
| declarations
|--------------
| constraints
--------------------------
```