UNIVERSITY OF WESTMINSTER

# ELECTRONICS AND COMPUTER SCIENCE

## 2010-2011

**Code:** 3SFE618

**Title:** Formal Methods

**Date:** 12 May 2011

**Time:** 14:00

**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 D.

Information sheets and additional stationery supplied:

# Section A

Answer ALL questions from this section

## Question 1

Given the following Z schemas:

$$
\begin{array}{|l}
\hline
\textit{First} \\
x, y : \mathbb{N} \\
A, B : \mathbb{P}\,\mathbb{N} \\
\hline
x < y \\
x \in A \\
A \subseteq B \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
\textit{Second} \\
z : \mathbb{N} \\
B, C : \mathbb{P}\,\mathbb{N} \\
\hline
z \in B \\
B \subseteq C \\
\hline
\end{array}
$$

**(a)** State and explain the formal definitions of $\Delta S$ and $\Xi S$ for any schema $S$. **[5 marks]**

**(b)** Give the *expanded* version of $\Delta \mathit{First}$. **[3 marks]**

**(c)** Give the *expanded* version of $\Xi \mathit{Second}$. **[4 marks]**

## Question 2

The following is part of a specification for the *Bikephone Shed* mobile phone shop. The following types representing individual phones, mobile manufacturers and unique mobile serial numbers.

$$
\begin{array}{ll}
MOBILE & ::= mobile1 \mid mobile2 \mid mobile3 \mid \ldots \\
MANUFACTURER & ::= Nokia \mid Apple \mid Blackberry \mid HTC \mid \ldots \\
SERIAL\_NO & ::= MP0012 \mid MP0008 \mid MP0099 \mid \ldots
\end{array}
$$

The state schema:

$$
\begin{array}{|l}
\hline
\textit{Bikephone\_Shed} \\
stock : \\
manufacturers : \\
serialnumb : \\
make : \\
price : \\
\hline
\quad \vdots \\
\hline
\end{array}
$$

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

- *stock* are the mobile phones waiting to be sold.
- *manufacturers* are the chosen phone manufacturers, e.g., Nokia, Apple, HTC, etc, whose phones *Bikephone Shed* sells.
- *serialnumb* maps each mobile to its *unique* serial number.
- *make* maps each mobile to its make.
- *price* maps each phone to price in pounds.

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

**[Continued Overleaf]**

**(b)** Define an **initial** *Bikephone Shed* state schema for the following situation:

- There are two phones $mobile1$ and $mobile2$.
- The chosen $manufacturers$ are Nokia, HTC and Blackberry.
- The registration numbers for $mobile1$ is $MP0012$ and for $mobile2$ is $MP0099$.
- The make of $mobile1$ is $Nokia$ and $mobile2$ is $HTC$.
- The price of each mobile is: $mobile1$ is £250 and $mobile2$ is £175. **[8 marks]**

## Question 3

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

Definition: The range restriction, $R \rhd A$, of the relation $R : X \leftrightarrow Y$ by the set $A : \mathbb{P}\, Y$ is the relation that relates $x$ to $y$, if and only if, $R$ relates $x$ to $y$ and $y$ is a member of the set $A$.

**[8 marks]**

## Question 4

The following:

$$\begin{aligned} multiply(3,6) &= 18 \\ subtract(24,9) &= 15 \\ increment(7) &= 8 \\ decrement(3) &= 2 \end{aligned}$$

are examples of the standard mathematical functions for *multipliction*, *subtraction*, *increment* and *decrement* respectively, for natural numbers ($\mathbb{N}$). Define the *signatures* of these functions. Note you are **not** required to give their definitions. **[10 marks]**

# Section B

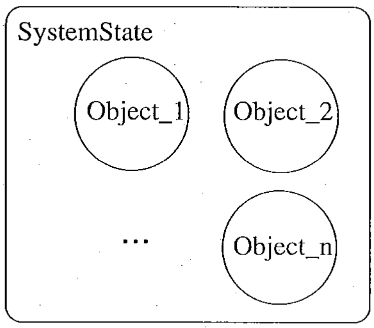
Answer TWO questions from this section

## Question 5

**(a)** When developing a Z specification of a system it is necessary to distinguish between *invalid*, *valid* and *initial* system states.

Describe the relationship between these different system states and explain the role of *system invariants* in this relationship. Illustrate this relationship using a diagram. **[7 marks]**

**(b)** Assume the general state of a system to be specified in Z consists of $n$ *objects* – $Object\_1$, .., $Object\_n$, as follows:




**(i)** Give a brief description of how this system's *general state schema* should be structured. Illustrate this structure by means of suitable Z schemas. **[6 marks]**

**(ii)** Give a brief description of how this system's *initial state schema* should be structured. Illustrate this structure by means of suitable Z schemas. **[7 marks]**

**[Continued Overleaf]**

**(iii)** The following operation is specified *only* in terms of $Object\_1$:

$$
\begin{array}{|l}
\hline
\textit{Operation\_on\_Object\_1} \\
\Delta\, Object\_1\_State \\
Input\,Variables \\
Output\,Variables \\
\hline
Pre\,Conditions \\
Object\_1' = New\_Object\_1 \\
Outputs \\
\hline
\end{array}
$$

Briefly explain how this operation could be re-used to define an operation on the *whole* of the system state using the technique known as *operation promotion*. Illustrate this by means of a diagram. In addition, give the Z schema that represents the promoted operation on the whole state. **[5 marks]**

## Question 6

The following is part of a Library specification.
The following definitions represent the set of books, copies (i.e. instances) of books and borrowers.

$[\, BOOK,\ COPY,\ BORROWER \,]$

$| \ maxloans : \mathbb{N}$

*LibraryDataBase*

$stock : COPY \rightarrowtail\!\!\!\mapsto BOOK$
$registeredborrowers : \mathbb{F}\, BORROWER$

*LibraryLoans*

$onloan : COPY \rightarrowtail\!\!\!\mapsto BORROWER$
$inlibrary : \mathbb{F}\, COPY$

---

$\forall\, b : BORROWER \bullet \#(onloan \rhd \{\, b \,\}) \leq maxloans$
$inlibrary \cap \operatorname{dom} onloan = \varnothing$

*Library*

*LibraryDataBase*
*LibraryLoans*

---

$\operatorname{dom} stock = inlibrary \cup \operatorname{dom} onloan$
$\operatorname{ran} onloan \subseteq registeredborrowers$

**[Continued Overleaf]**

*IssueBook*

$$\Delta Library$$
$$c? : COPY$$
$$b? : BORROWER$$

---

$$c? \in inlibrary$$
$$stock' = stock$$
$$inlibrary' = inlibrary \setminus \{ c? \}$$
$$\#(onloan \rhd \{ b? \}) < maxloans$$
$$registeredborrowers' = registeredborrowers$$
$$b? \in registeredborrowers$$
$$onloan' = onloan \oplus \{ c? \mapsto b? \}$$

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

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

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

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

**(c)** Specify the *ReturnBook* operation which is used when a borrower returns a book to the library. 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. **[12 marks]**

## Question 7

Part of a Z specification for a University's Computing degree course in particular module registration and de-registration is given in Appendix A.

**(a)** The ZTC type checker output for the Module specification is given in Appendix B. For each error reported give an explanation and the necessary corrections. **[10 marks]**

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

**(c)** Assuming all of the necessary modifications required in part **(b)** have been made, an attempt is made to animate the Module specification in ZANS.

However, ZANS indicates that it can not animate the two operations `RegisterForModule_Ok` and `DeregisterFromModule_Ok`, by reporting that they are *"not explicit"*.

Give a brief explanation of what *"not explicit"* means in this context and state what modifications need to be made to these two operation schemas to correct this problem. **[5 marks]**

# Appendix A. Module Specification

## A.1 ZTC Box Style Version

```
specification

[ STUDENT ]

TITLE   ::= Formal_Methods | Compilers | Networks

STAFF   ::= P_Howells | A_Lecturer | M_Mouse

REPORT  ::= Success
          | ERROR_Already_Registered
          | ERROR_Module_Full
          | ERROR_Not_Registered

| MODULE_LIMIT : N

| ACADEMIC_STAFF : P STAFF

--- Module -------------------------------
| title : TITLE ;
| leader : STAFF ;
| numbstudents : N ;
| students : P STUDENT
|--------------------------------
| leader in ACADEMIC_STAFF ;
| numbstudents <= MODULE_LIMIT ;
| numbstudents = students
-------------------------------------------


--- InitialModule -------------------------
| Module
|--------------------------------
| title = Formal_Methods ;
| leader = P_Howells ;
| numbstudents = 0 ;
| students = {}
-------------------------------------------
```

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

--- RegisterForModule_Ok --------------
|  Delta Module ;
|  newstudent? : STUDENT
|--------------------------------
|  newstudent? notin students ;
|  numbstudents < MODULE_LIMIT ;
|  students' = students || newstudent? ;
|  numbstudents' = numbstudents + 1 ;
-------------------------------------------

RegisterForModule_Success =^= RegisterForModule_Ok
                                /\ ReportSuccess

--- AlreadyRegistered_Error -----------
|  Xi Module ;
|  newstudent? : STUDENT ;
|  report! : REPORT
|--------------------------------
|  newstudent? in students
|  report! = ERROR_Already_Registered
----------------------------------------------

--- ModuleFull_Error ----------------------
|  Xi Module ;
|  newstudent? : STUDENT ;
|  report! : REPORT
|--------------------------------
|  numbstudents = MODULE_LIMIT ;
|  report! = ERROR_Module_Full
-------------------------------------------

RegisterForModule  =^= RegisterForModule_Success
                       \/ AlreadyRegistered_Error
                       \/ ModuleFull_ERROR
```

```
--- DeregisterFromModule_Ok ---------
|  Module ;
|  deregstudent? : STUDENT
|--------------------------------
|  deregstudent? in students ;
|  students' = students \ { deregstudent? } ;
|  numbstudents' = numbstudents - 1 ;
-------------------------------------------

DeregisterFromModule_Success
        =^= DeregisterFromModule_Ok /\ ReportSuccess

--- Student_Not_Registered_Error ----------
|  Xi Module ;
|  deregstudent? : STUDENT ;
|  report! : REPORT
|--------------------------------
|  deregstudent? = students ;
|  report! = ERROR_Not_Registered
-------------------------------------------

DeregisterFromModule
        =^= DeregisterFromModule_Success \/
                Student_Not_Registered_Error

--- PlacesLeftOnModule_Ok ---------------------
|  Xi Module ;
|  placesleft! : N
|--------------------------------
|  placesleft = MODULE_LIMIT - numbstudents
-----------------------------------------------

PlacesLeftOnModule =^= PlacesLeftOnModule_Ok /\
                        ReportSuccess
```

## A.2 Pretty-Printed Version

$[STUDENT]$

$TITLE ::= Formal\_Methods \mid Compilers \mid Networks$

$STAFF ::= P\_Howells \mid A\_Lecturer \mid M\_Mouse$

$$
\begin{array}{ll}
REPORT ::= & Success \\
 & \mid ERROR\_Already\_Registered \\
 & \mid ERROR\_Module\_Full \\
 & \mid ERROR\_Not\_Registered
\end{array}
$$

$$| \; MODULE\_LIMIT : \mathbb{N}$$

$$| \; ACADEMIC\_STAFF : \mathbb{P}\, STAFF$$

**Module**

$$
\begin{array}{|l}
title : TITLE \\
leader : STAFF \\
numbstudents : \mathbb{N} \\
students : \mathbb{P}\, STUDENT \\
\hline
leader \in ACADEMIC\_STAFF \\
numbstudents \leq MODULE\_LIMIT \\
numbstudents = students
\end{array}
$$

**InitialModule**

$$
\begin{array}{|l}
Module \\
\hline
title = Formal\_Methods \\
leader = P\_Howells \\
numbstudents = 0 \\
students = \varnothing
\end{array}
$$

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

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{RegisterForModule\_Ok}} \\
\Delta Module \\
newstudent? : STUDENT \\
\hline
newstudent? \notin students \\
numbstudents < MODULE\_LIMIT \\
students' = students \cup newstudent? \\
numbstudents' = numbstudents + 1 \\
\hline
\end{array}
$$

$$RegisterForModule\_Success \mathrel{\widehat=} RegisterForModule\_Ok \land ReportSuccess$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{AlreadyRegistered\_Error}} \\
\Xi Module \\
newstudent? : STUDENT \\
report! : REPORT \\
\hline
newstudent? \in students \\
report! = ERROR\_Already\_Registered \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{ModuleFull\_Error}} \\
\Xi Module \\
newstudent? : STUDENT \\
report! : REPORT \\
\hline
numbstudents = MODULE\_LIMIT \\
report! = ERROR\_Module\_Full \\
\hline
\end{array}
$$

$$
\begin{aligned}
RegisterForModule \mathrel{\widehat=}\ & RegisterForModule\_Success \\
& \lor AlreadyRegistered\_Error \\
& \lor ModuleFull\_ERROR
\end{aligned}
$$

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{DeregisterFromModule\_Ok}} \\
Module \\
deregstudent? : STUDENT \\
\hline
deregstudent? \in students \\
students' = students \setminus \{deregstudent?\} \\
numbstudents' = numbstudents - 1 \\
\hline
\end{array}
$$

$$\begin{aligned}DeregisterFromModule\_Success \mathrel{\widehat{=}}\ & DeregisterFromModule\_Ok \\ & \land\ ReportSuccess\end{aligned}$$

$$\begin{array}{|l}\hline \textit{Student\_Not\_Registered\_Error} \\ \Xi Module \\ deregstudent? : STUDENT \\ report! : REPORT \\ \hline deregstudent? = students \\ report! = ERROR\_Not\_Registered \\ \hline \end{array}$$

$$\begin{aligned}DeregisterFromModule \mathrel{\widehat{=}}\ & DeregisterFromModule\_Success \\ & \lor\ Student\_Not\_Registered\_Error\end{aligned}$$

$$\begin{array}{|l}\hline \textit{PlacesLeftOnModule\_Ok} \\ \Xi Module \\ placesleft! : \mathbb{N} \\ \hline placesleft = MODULE\_LIMIT - numbstudents \\ \hline \end{array}$$

$$PlacesLeftOnModule \mathrel{\widehat{=}} PlacesLeftOnModule\_Ok \land ReportSuccess$$

## Appendix B. ZTC output for Module Specification

The following is part of the ZTC type checker output from the **Module** specification.

```
Parsing main file: module.zbx
... Type checking Given set. "module.zbx" Line 3
... Type checking Free type definition: TITLE. "module.zbx" Line 5
... Type checking Free type definition: STAFF. "module.zbx" Line 7
... Type checking Free type definition: REPORT. "module.zbx" Lines 9-12
... Type checking Axiom box. "module.zbx" Line 14
... Type checking Axiom box. "module.zbx" Line 16
... Type checking Schema box: Module. "module.zbx" Lines 18-26
--- Typing error. "module.zbx" Line 26. Type mismatch:
... Type checking Schema box: InitialModule. "module.zbx" Lines 29-35
... Type checking Schema definition: ReportSuccess. "module.zbx" Line 38
... Type checking Schema box: RegisterForModule_Ok. "module.zbx" Lines 40-47
--- Typing error. "module.zbx" Line 46. Type mismatch: Infix expression
--- Typing error. "module.zbx" Line 46. Type mismatch: Right-hand side
... Type checking Schema definition: RegisterForModule_Success. "module.zbx"
                                      Lines 50-51
... Type checking Schema box: AlreadyRegistered_Error. "module.zbx" Lines 53-59
--- Typing error. "module.zbx" Line 58. Mapping expected:
... Type checking Schema box: ModuleFull_Error. "module.zbx" Lines 62-68
--- Syntax error. "module.zbx" Line 73, near "ModuleFull_ERROR"
... Type checking Schema box: DeregisterFromModule_Ok. "module.zbx" Lines 75-81
--- Typing error. "module.zbx" Line 80. Undefined name: students'
--- Typing error. "module.zbx" Line 81. Undefined name: numbstudents'
... Type checking Schema definition:DeregisterFromModule_Success. "module.zbx"
                                     Lines 84-85
... Type checking Schema box: Student_Not_Registered_Error. "module.zbx" Lines 87-93
--- Typing error. "module.zbx" Line 92. Type mismatch:
... Type checking Schema definition: DeregisterFromModule. "module.zbx" Lines 96-98
... Type checking Schema box: PlacesLeftOnModule_Ok. "module.zbx" Lines 100-104
--- Typing error. "module.zbx" Line 104. Undefined name: placesleft
... Type checking Schema definition: PlacesLeftOnModule. "module.zbx" Lines 107-108
--- Reached the end of the main file while parsing.
End of main file: module.zbx
```

## Appendix C. ZANS output for Plane Specification

The following is part of the ZANS animator output from the **Plane** specification.

```
zans> load plane.zbx
zans> animate
... Initialization.
    Looking for state-schema pragma.
    No state-schema pragma found. Attempt auto-set.
    State schema: Plane
    State schema: InitialPlane
... Analyzing axiomatic definitions: not explicit.
... Analyzing schema Plane: state.
... Analyzing schema InitialPlane: state.
... Analyzing schema Board_OK: explicit.
... Analyzing schema ReportSuccess: explicit.
... Analyzing schema Board_Succ: explicit.
... Analyzing schema Board_AlreadyOnBoard: explicit.
... Analyzing schema Board_PlaneFull: explicit.
... Analyzing schema Board: explicit.
... Analyzing schema Disembark_OK: explicit.
... Analyzing schema Disembark_Succ: explicit.
... Analyzing schema Disembark_NotOnBoard: explicit.
... Analyzing schema Disembark_PlaneEmpty: explicit.
... Analyzing schema Disembark: explicit.
... Analyzing schema NumberOnBoard: explicit.
... Analyzing schema PersonOnBoard: explicit.
... Initializing global names.
capacity:  Undef
... Check initialization schemas.
    Looking for init-schema pragma.
    No init-schema pragma found. Attempt auto-set.
Initialization schema: ReportSuccess
... Initializing ReportSuccess
... Execute schema: ReportSuccess
Schema: ReportSuccess
report!:  OK

anim> exit
```

## Appendix D. Table of Z Syntax

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

### D.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 $S$ |

## D.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 $P$ |
| $\exists_1 x : T \bullet P$ | There exists a *unique* element $x$ of type $T$ which satisfies $P$ |

## D.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 |

## D.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 $Y$ |
| dom $R$ | Domain of relation $R$ |
| ran $R$ | Range of relation $R$ |
| $S \lhd R$ | Domain restriction of $R$ to the set $S$ |
| $S \mathbin{⩤} R$ | Domain anti-restriction of $R$ by the set $S$ |
| $R \rhd S$ | Range restriction of $R$ to the set $S$ |
| $R \mathbin{⩥} S$ | 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 $R$ |
| id $X$ | Identity relation |
| $R^{-1}$ | Inverse relation |
| $R^{+}$ | Transitive closure of $R$ |
| $R^{*}$ | Reflexive-transitive closure of $R$ |

## D.5 Functions

| Notation | Description |
|---|---|
| $\twoheadrightarrow\!\!\!\!\!\!\!+\!\!+$ | Finite function |
| $\rightarrowtail\!\!\!\!\!\!\!+\!\!+$ | Finite injection |
| $\nrightarrow$ | Partial function |
| $\rightarrow$ | Total function |
| $\rightarrowtail\!\!\!\!\!\!+$ | Partial injection |
| $\rightarrowtail$ | Total injection |
| $\twoheadrightarrow\!\!\!\!\!\!+$ | Partial surjection |
| $\twoheadrightarrow$ | Total surjection |
| $\rightarrowtail\!\!\!\!\!\rightarrow\!\!\!\!\twoheadrightarrow$ | Bijection |

## D.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 $t$ |

## D.7 Bags

| Notation | Description |
|---|---|
| $\text{bag}\, X$ | Bag of type $X$ |
| $[\![\,]\!]$ | Empty bag |
| $x \sqsubset\!\!\!- B$ | Bag membership |
| $x \not\sqsubset\!\!\!- B$ | 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 |
| $B_1 \uminus B_2$ | 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 $s$ |

## D.8 Schemas

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

## D.9 Schema Calculus

| **Notation** | **Description** |
|---|---|
| $[S;\ declarations \mid constraints]$ | Schema inclusion |
| $S'$ | Schema decoration |
| $\Delta S$ | $\Delta$ (Delta) Convention |
| $\Xi S$ | $\Xi$ (Xi) Convention |
| $S \wedge T$ | Schema Conjunction ($S$ and $T$) |
| $S \vee T$ | Schema Disjunction ($S$ or $T$) |