Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (0223) 323010**
INTERNATIONAL: **+44 223 323010**
FAX: **+44 223 359779**
E-MAIL: **apm@ansa.co.uk**

ANSA Phase III

Z and Object Z Notations in FrameMaker Documents

Alastair J Tocher and Gray Girling

Abstract

TThis document incorporates a set of templates suitable for the representation of Z and Object Z specification boxes: axiomatic definitions, schemas, generic definitions, and class definitions.

It is substantially based on an original document by Alastair J Tocher of BNR Europe Limited (Software and Systems Engineering).

Feedback is welcomed.

APM.1267.00.01 **Draft** 21 June 1994

Request for Comments (confidential to ANSA consortium for 2 years)

Distribution:
Supersedes:
Superseded by:

Contents

1 Introduction

1.1 Purpose

This document incorporates a set of templates suitable for the representation of Z and Object Z specification boxes: axiomatic definitions, schemas, generic definitions, and object class definitions. The templates offered are intended to make entry of Z and Object Z specifications to FrameMaker documents both faster and syntactically less error-prone.

Feedback, both on these templates and on the entry of Z and Object Z notations in FrameMaker, is welcomed.

1.2 Overview

The document is presented in the sections addressing

- the definition of and means of accessing both the Z box templates and the relevant text formats,
- guidance on the use of the boxes and formats to obtain best use of them,
- schema notations for expressing states and operations,
- linear notations for expressing
  - — definitions and declarations,
  - — logic,
  - — numbers,
  - — relations,
  - — functions,
  - — orders and equivalances,
  - — sequences, and
  - — bags.

1.3 Background

The reader is assumed to be familiar with the principles of Z notation, of Object Z notation, and of FrameMaker document features.

2 Templates

Z box templates are offered as FrameMaker tables. Entry of data expands the table cells, thereby extending the Z schema boxes to fit the text included. The following kinds of Z box are supported:

- Z axiomatic definition boxes
- Z schema definition boxes
- Z generic definition boxes
- Object Z class boxes

These are supported by the following FrameMaker table formats, available from the “Insert Table” option within the “Table” menu:

- Z_Box_Axiomatic
- Z_Box_Schema
- Z_Box_Generic
- Z_Box_Class_Header, Z_Box_Class_Variables, and Z_Box_Class_Footer

Three new paragraph formats are also provided for use in conjunction with these Z boxes:

Z_Box_Name:
Used for the box names.

Z_Box_Body:
Used both for the declaration part and the axiom part of the Z axiomatic, schema, generic, and class variables boxes. Tab stops are provided in Z_Box_Body including a final right-aligning tab stop which is provided for brief labelling/commenting of axioms if desired (example given below).

Z_Box_Class_Body:
Used as for Z_Box_Body, but in Z_Box_Class_Header (to give better text alignment than would be given by Z_Box_Body).

2.1 Z axiomatic definition box

An example expanding box suitable for Z axiomatic definitions.

limit: NAT
limit ≤ 65535 [short comments here]

2.2 Z schema definition box

An example expanding box suitable for Z schemas.

The following linear syntax is also available:

COUNTER
value: NAT
value $\leq$ limit

COUNTER $\triangleq$ [value: $\mathbb{N}$ | value $\leq$ limit]

2.3 Generic Z schema definition box

An example expanding box suitable for a generic Z schemas.

COUNTERSET[T]
values: $\mathbb{P}$(T) limit: T
$\forall$value: values $\bullet$ value $\leq$ limit

The following linear syntax is also available:

COUNTERSET[T] $\triangleq$ [values: P(T); limit: T | $\forall$value: values $\bullet$ value $\leq$ limit]

2.4 Z generic definition box

An example expanding box suitable for Z generic definitions.

Pool [RESOURCE]
owner: RESOURCE $\nrightarrow$ USER free: **set** RESOURCE
(**dom** owner) $\cup$ free = RESOURCE (**dom** owner) $\cap$ free = $\emptyset$

The following linear syntax is also available:

Dictionary[X] == Pool[X]

2.5 Object Z class definition box

The object Z class box template comes in three parts: the header, the variable declarations, and the footer, and may be used in conjunction with the existing Z boxes.

The header supports capture of the class name, the visibility list, inherited classes, and constants; the variable declarations supports entry of the variables local to a class; the footer serves only as the end marker of the class definition.

An example expanding box suitable for a Z class header.

Queue[T]	
‡(Init, Join, Leave)	
MaxQueueSize:	NAT_1

An example expanding box suitable for Z class variable declarations.

items: **seq** T

#items ≤ MaxQueueSize

An example box suitable for a Z class footer.

Putting these together with schemas denoting some operations gives the following example.

Queue [T]

‡(Init, Join, Leave)
MaxQueueSize: NAT_1

items: **seq** T

#items ≤ MaxQueueSize

Init

items = <>

Join

Δ(items)
in?: T

#items < MaxQueueSize
items' = items^<in?>

Leave

Δ(items)
out!: T

items ≠ <>
items = <out>^items'

Note: This example also makes use of cross-references to show the exported operations names in the Class header: renaming the operations themselves and updating the fields will rename in the export list too. It is not obvious whether this will be a useful facility in the long term.

3 Usage tips

3.1 Boxes with only declarations or only axioms

In some cases, Z specifications will only have either declarations or only axioms. In such cases, insert an appropriate Z box, then select one row of the FrameMaker table and delete that row (a Framemaker prompt will ask whether the cell contents or the cells themselves are to be deleted: select "remove cells"). Fill in the remaining cell as normal.

An example is shown in operation Init above.

3.2 Tabs in specifications

To enter tabs in a Z box, type ESC-TAB – a TAB character alone will move the text entry point to the next box cell or to the title of the box as appropriate.

3.3 Avoiding splitting Z boxes over page boundaries

The table formats provided do not automatically provide for boxes which might be split over a page boundary. To prevent a box from being split over a boundary, the user must manually select the top cell of the box (i.e. the top cell of the FrameMaker table) and, selecting the "Row Format" option from the "Table" menu, set the row properties to "Keep with next row".

Note: It does not seem possible to set this as a default option in the basic box templates. If any reader knows how to achieve this, please contact the author of this document.

3.4 Finding and referring to Z boxes

To find, or to generate a reference to, a named Z box (schema or generic definition), select the "Cross reference" option from the "Special" menu.

Then select paragraph type "Z_Box_Name" within the offered menu box. The names of the boxes will then be displayed in the menu box. Select one and then select the appropriate option (Go To, Generate Reference, etc.).

If "Go To" is selected, the selected box will be displayed on the screen as the current page.

4 Types

[A, B]	Introduction of generic sets
$T_1 \times T_2 \times ... \times T_n$	The Cartesian (or cross) product of T_1, T_2, ..., T_n: the set of n-tuples such that the j^{th} element is of type T_j.
$\mathbb{P}$ X **set** X	The powerset (i.e. the set of all subsets) of X
nonemptyset X	The set of all non-empty subsets of X: $\equiv \{ x: \mathbb{P}\ X \mid x \neq \emptyset \}$
$\mathbb{F}$ X **finiteset** X	The set of all finite subsets of X: $\equiv \{ T: \mathbb{P}\ X \mid T$ *is finite* $\}$
$x_1 \langle\langle T_1 \rangle\rangle \mid x_2 \langle\langle T_2 \rangle\rangle \ldots \mid x_n \langle\langle T_n \rangle\rangle \mid a_1 \mid a_2 \ldots \mid a_n$	A combined disjoint union of types T_1, T_2, … T_n mapped by functions x_1, x_2, x_n with the enumerated type a_1, a_2 … a_n
$\lhd\ x_1{:}T_1; ... ; x_n{:}T_n \rhd$	Schema type definition (meta - Z)

5 Schema calculus

The symbols used for schema calculus generally use a larger font size in order to try to distinguish them from the analogous logical operators. Let S and T be schema types and P be a predicate.

QS	Schema instance: a schema tuple of the same schema type as S each of the fields of which have the value associated with variables (that must be defined at the point it is used) with the same names as the field names in S.
pred S	The predicate part of a schema.
$S \setminus (v_1, v_2, \dots v_n)$ $S \setminus v_1$	Hiding: the schema without $v_1, v_2, \dots v_n$ in its variable list with a predicate **pred** S in which variables $v_1, v_2, \dots v_n$ are existentially quantified.
S \ P	Schema hiding: schema S in which the variables of P are hidden.
S | P	The schema with the signature of S and predicate **pred** S ∧ P.
¬ S	Schema S with negated predicate ¬**pred** S.
S ∨ T	A schema with merged S and T signatures with the predicate formed by the siajunction of S and T predicates **pred** S ∨ **pred** S
S ∧ T	A schema with merged S and T signatures with the predicate formed by the conjunction of S and T predicates **pred** S ∧ **pred** S
S ⇒ T	Schema implication: “S implies T” ≡ ((¬ S) ∨ T)
S ⇔ T	Schema equivalence: “S is logically equivalent to T” ≡ (S ⇒ T) ∧ (T ⇒ S)
S', S^i, S_n, etc	Decoration indicating the schema resulting from the systematic renaming (with a prime, subscript, superscript etc.) of variables defined in a schema
Δ S	The schema with all the declarations and axioms in X, together with S’ ≡ S ∧ S’
Ξ S	The schema with all the declarations and axioms in S together with S’ with a predicate that identifies the value of each dashed variable with the value of its undashed counterpart: ≡ S ∧ S’ | ΘS = ΘS’

6 Definitions and declarations

Let x, xi be identifiers, and let T, T_i be sets:

LHS == RHS LHS ≡ RHS	Definition of LHS as syntactically equivalent to RHS.
S ≙ [<schema expression>]	Definition of schema S
T ::= x_1«T_1» | x_2«T_2» ... | x_n«T_n» | a_1 | a_2 ... | a_n	Declare T as a combined disjoint union of types T_1, T_2, … T_n mapped by functions x_1, x_2, x_n with the enumerated type a_1, a_2 … a_n (the declaration may be recursive)
x: T	Declaration of x to be of type T
x_1:T_1; x_2:T_2; ... ; x_n:T_n	List of declarations
x_1, x_2, ... , x_n:T	List of declarations: ≡ x_1:T; x_2:T; ... ; x_n:T

7 Logic

Let P, Q be predicates, and let D be a declaration:

(P)	Parenthesis: $\equiv$ P
true, **false**	Logical constants
$\neg$ P	Negation of predicate P: “not P”
P $\vee$ Q	Disjunction of predicates P and Q: “P or Q”
P $\wedge$ Q	Conjunction of predicates P and Q: “P and Q”
P $\Rightarrow$ Q	Implication: “P implies Q” $\equiv ((\neg P) \vee Q)$
P $\Leftarrow$ Q	Reverse implication: “P is implied by Q” $\equiv Q \Rightarrow P$
P $\Leftrightarrow$ Q	Equivalence: “P is logically equivalent to Q” $\equiv (P \Leftarrow Q) \wedge (P \Rightarrow Q)$
P « Q » R	Conditional: “P if Q otherwise R”
$\forall$x:T $\bullet$ P	Universal quantification: “for all x in type T, P holds”
$\exists$x:T $\bullet$ P	Existential quantifiction: “there exists at least one x in type T such that P holds
$\exists$**!** x:T $\bullet$ P_x	Uniqueness of existence: “there exists precisely one x in type T such that P_x holds” $\equiv \exists\ x{:}T \bullet (P_x \wedge \forall y{:}T \bullet (P_y \Rightarrow x{=}y))$
$\forall$D | P $\bullet$ Q	$\equiv \forall D \bullet P \Rightarrow Q$
$\exists$D | P $\bullet$ Q	$\equiv \exists D \bullet P \wedge Q$
s = t	Equality
s $\neq$ t	Inequality $\equiv \neg (s = t)$

8 Sets

$t \in S$	Set membership: "t is a member of S"
$t \notin S$	Set non-membership: "t is not a member of S" $\equiv \neg (t \in S)$
$S \subseteq T$	Subset: "S is a subset of T" $\equiv (\forall x{:}S \bullet t \in S)$
$S \supseteq T$	Superset: "S is a superset of T' $\equiv T \subseteq S$
{}	The empty set.
	Alternative notation for the empty set $\equiv \{\}$
$\{x_1, x_2, \ldots, x_n\}$	The set containing precisely the members x_1, .. , and x_n.
$\{ x{:}T \mid P \}$	The set containing all and only all those x in type T for which P holds.
$\{ D \mid P \bullet t \}$	The set of t's such that, given the declarations D, P holds.
$\{ D \bullet t \}$	$\equiv \{ D \mid \mathbf{true} \bullet t \}$
$S \cap T$	Set intersection: for S, T: **set** X, $\equiv \{ x{:} X \mid x \in S \land x \in T \}$
$S \cup T$	Set union: for S, T:**set** X, $\equiv \{ x{:} X \mid x \in S \lor x \in T \}$
$S - T$	Set difference:for S, T:**set** X, $\equiv \{ x{:} X \mid x \in S \lor x \notin T \}$
$\bigcap SS$	Distributed set intersection: for SS: **set** (**set** X) $\equiv \{ x{:} X \mid (\forall S{:}SS \bullet x \in S) \}$
$\bigcup SS$	Distributed set union:for SS: **set** (**set** X) $\equiv \{ x{:} X \mid (\exists S{:}SS \bullet x \in S) \}$
#S	The number of distinct elements in the finite set S:
disjoint SS	The set, SS, of sets is pairwise disjoint: $\equiv \forall S_1,S_2{:}SS \bullet (S_1 \cap S_2 \neq \emptyset) \Leftrightarrow S_1 = S_2$
partition S	The set of all partitions of S: $\equiv \{ SS{:} \mathbb{P} (\mathbb{P} S)) \mid \mathbf{disjoint}\ SS \land \bigcup SS = S \}$

9 Numbers

NAT $\mathbb{N}$	The set of Natural Numbers
NAT_1 $\mathbb{N}_1$	The set of strictly positive Natural Numbers: $\equiv \{ n{:}NAT \mid n > 0 \}$
INT $\mathbb{Z}$	The set of Integers
m .. n	The set of Integers from m to n inclusive: $\equiv \{ k{:}INT \mid m \leq k \wedge k \leq n \}$

10 Relations

A relation is represented by a set of ordered pairs. Therefore all operators defined over sets may also be applied freely to relations.

Note that a consistant and easy to understand symbology for functions and relations is used in the following proposed by A J Tocher consisting of the use of normal printable characters <, >, - and =. However, these are not yet widely recognized symbols and so their more generally used equivalents (where they exist) are listed first and should probably be preferred in documents for public dissemination. The notation splits a function or relation symbol into three areas corresponding to domain coverage (< or <<) mapping type (==, -=, =-, or --), and range coverage (> or >>).

Domain coverage:

<<	every element in the domain is present in at least one tuple (the mapping is “total”)
<	some elements in the domain may not be involved in a tuple

Range coverage:

>>	every element in the range is present in at least one tuple (the mapping is “on to”)
>	some elements in the range may not be involved in a tuple

Mapping type:

==	the mapping is many to many
-=	the mapping is one to many
=-	the mapping is many to one (a function)
--	the mapping is one to one (an “injection” function)

Letting X, Y and Z be such sets such that x:X, y:Y, and R, R1, R2: $X \leftrightarrow Y$:

$X \leftrightarrow Y$ X <==> Y	The set of all relations from X to Y: $\equiv \mathbb{P}\,(X \times Y)$
x R y	x is related by R to y: $\equiv (x, y) \in R$
$x \mapsto y$	$\{x \mapsto y\} \in R \Leftrightarrow x\ R\ y$
[]	The empty relation $\equiv \emptyset$
dom R	The domain of relation R:{ $\equiv \{\ x{:}\ X \mid (\exists y{:}Y \bullet x\ R\ y\ )\ \}$

ran R **rng** R	The range of relation R: $\equiv \{ y{:}Y \mid (\exists x{:}X \bullet x\ R\ y) \}$
$R_1 ; R_2$	Forward relational composition: for $R_1: X \leftrightarrow Y$ and $R_2: Y \leftrightarrow Z$ $\equiv \{ x{:}\ X;\ z{:}\ Z \mid (\exists y{:}Y \bullet x\ R1\ y \wedge y\ R_2\ z) \}$
$R_1 \circ R_2$	Relational composition: $\equiv R_2 ; R_1$
R^{-1}	Inverse of relation R: $\equiv \{ y{:}\ Y;\ x{:}\ X \mid x\ R\ y \}$
id X	Identity relation on set X: $\equiv \{ x, y{:}\ X \mid x = y \}$
R^k	Relation R composed with itself k times: $R^0 \equiv \mathbf{id}\ X$ $R^{k+1} \equiv R ; R^k$
R^*	Reflexive, transitrive closure of relation R: $\equiv \bigcup \{ n{:}NAT \}$
R^+	Non-reflexive transitive closure of relation R: $\equiv \bigcup \{ n{:}NAT_1 \bullet R^n \}$
$S \lhd R$ S **resd** R	Restriction of domain to S: $\equiv \{ (x, y){:}\ R \mid x \in S \}$
$S ⩤ R$ S **subd** R	Subtraction of S from domain: $\equiv (X - S) \lhd R$
$R \rhd S$ R **resr** S	Restriction of range to S: $\equiv \{ (x, y){:}\ R \mid y \in S \}$
$R ⩥ S$ R **subr** S	Subtraction of S from range: $\equiv R \rhd (Y - S)$
$R_1 \oplus R_2$	R_1 overwritten by R_2: $\equiv \{ (x, y){:}\ R_1 \mid x \notin \mathbf{dom}\ R_2 \} \cup R_2$
R ⦇ S ⦈	Relational image of S in R $\equiv \{ y{:}\ \mathbf{ran}\ R \mid (\exists x{:}\mathbf{S} \bullet (x, y) \in R) \}$
X <<==> Y	The set of total relations from X to Y: $\equiv \{ R{:}\ X \leftrightarrow Y \mid \mathbf{dom}\ R = X \}$
X <==>> Y	The set of all onto relations from X to Y: $\equiv \{ R{:}\ X \leftrightarrow Y \mid \mathbf{ran}\ R = Y \}$
X <<==>> Y	The set of all total onto relations from X to Y: ≡ (X <<==> Y) ∩ (X <==>> Y)
X <-=> Y	The set of all one-many relations from X to Y: $\equiv \{ R{:}\ X \leftrightarrow Y \mid (\forall y{:}\ \mathbf{ran}\ R \bullet (\exists!\ x{:}X \bullet x\ R\ y)) \}$
X <<-=> Y	The set of all total one-many relations from X to Y: ≡ (X <<==> Y) ∩ (X <-=> Y)

X <-=>> Y	The set of all one-many onto relations from X to Y: ≡ (X <−=> Y) ∩ (X <==>> Y)
X <<-=>> Y	The set of all total one-many onto relations from X to Y: ≡ (X <<−=> Y) ∩ (X <==>> Y)

11 Functions

Functions are many-one relations. Therefore all operators defined over relations may also be applied to functions.

X ⇸ Y X <=-> Y	The set of all functions (i.e. many-one relations) from X to Y: ≡ { R: X ↔ Y | (∀x: **dom** R • (∃**!** y:Y • x R y)) }
X → Y X <<=-> Y	The set of all total functions from X to Y: ≡ (X ⇸ Y) ∩ (X <<==> Y)
X <=->> Y X ⤀ Y	The set of all onto functions from X to Y: ≡ (X ⇸ Y) ∩ (X <==>> Y)
X ↠ Y X <<=->> Y	The set of all total onto functions from X to Y: ≡ (X → Y) ∩ (X <==>> Y)
X ⤔ Y X <--> Y	The set of all one-one functions from X to Y: ≡ (X ⇸ Y) ∩ (X <−=> Y)
X ↣ Y X <<--> Y	The set of all total one-one functions from X to Y: ≡(X ⤔ Y) ∩ (X <<==> Y)
X ⤗ Y X <-->> Y	The set of all one-one onto functions from X to Y: ≡ (X ⤔ Y) ∩ (X <==>> Y)
X ⤖ Y X <<-->> Y	The set of all total one-one onto functions from X to Y: ≡ (X ↣ Y) ∩ (X <==>> Y)
F x	Function application: ≡ the unique y such that x F y
(λx:X | P • t)	Lambda abstraction: the function that, given an argument of type X such that P holds, returns the result t (i.e. it relates x to t): ≡ {x:X | P • (x, t) }
(λx:X • t)	≡ (λx:X | **true** • t)
$(\lambda x_1{:}X_1, x_2{:}X_2, \ldots, x_n{:}X_n \mid P \bullet t)$	$\equiv \{x_1{:}X_1, x_2{:}X_2, \ldots, x_n{:}X_n \mid P \bullet ((x_1, x_2, \ldots, x_n), t)\}$

12 Orders and equivalences

transitiveorder X	The set of transitive orders on X; $\equiv$ {R: X $\leftrightarrow$ X $\mid \forall x_1, x_2, x_3{:}X \bullet (x_1\ R\ x_2 \wedge x_2\ R\ x_3) \Rightarrow x_1\ R\ x_3$ }
preorder X	The set of preorders (transitive, reflexive) on X: $\equiv$ {R: **transitiveorder** X $\mid \forall x{:}X \bullet x\ R\ x$ }
partialorder X	The set of partial orders (transitive, reflexive, anti-symmetric) on X: $\equiv$ {R: **preorder** X $\mid \forall x_1, x_2{:}X \bullet (x_1\ R\ x_2 \wedge x_2\ R\ x_1) \Rightarrow x_1 = x_2$ }
totalorder X	The set of total orders (reflexive, transitive, anti-symmetric, and ...) on X: $\equiv$ {R: **partialorder** X $\mid \forall x_1, x_2{:}X \bullet (x_1\ R\ x_2 \vee x_2\ R\ x_1)$ }
equivalence X	The set of equivalence relations (reflexive, transitive, symmetric) on X: $\equiv$ {R: **preorder** X $\mid \forall x_1, x_2{:}X \bullet (x_1\ R\ x_2 \Rightarrow x_2\ R\ x_1)$ }

13 Sequences

seq X	The set of sequences, all of whose members are in set X: $\equiv \{A{:}NAT_1 \leftrightarrow X \mid \exists n{:}NAT \bullet \mathbf{dom}\ A = 1 .. n\}$
<>	The empty sequence: $\equiv \varnothing$
$<a_1, a_2, ..., a_n>$	$\{(1,a_1), (2,a_2), ... , (n,a_n)\}$
$<a_1, a_2, ..., a_n>$^$<b_1, b_2, ..., b_m>$	Catenation: $\equiv <a_1, a_2, ..., a_n, b_1, b_2, ..., bm>$
elems X	The set of elements which appear in the sequence: ≡ **ran** S
head X	The first element in the sequence: ≡ S(1)
last X	The final element of a non-empty sequence: ≡ S(#S)

14 Bags

bag X	The set of bags whose elements are drawn from set X:
⟦⟧	The empty bag: ≡ ∅
⟦ x_1, x_2, ... , x_n ⟧	The bag containing x_1, ..., x_n with the frequency with which they appear in the list.
items S	The bag of items contained in the sequence S: ≡ { x: **ran** S • (x, #{ i: **dom** S | S(i) = x }) }