# 1 Z Notation Introduction

The following subsections provide a high level overview of select properties of Z Notation based on "The Z Notation: A Reference Manual" by J. M. Spivey. A copy of this reference manual can be found at dave/docs/z/Z-notation reference manual.pdf. In many cases, definitions will be pulled directly from the reference manual and when this occurs, the relevant page number(s) will be included. For a proper introduction with tutorial examples, see chapter 1, "Tutorial Introduction" from pages 1 to 23. For the $LaTeX$ symbols used to write Z, see the reference document found at dave/docs/z/zed-csp-documentation.pdf.

## 1.1 Decorations

The following decorations are used throughout this document and are taken directly from the reference manual. For a complete summary of the Syntax of Z, see chapter 6, Syntax Summary, starting on page 142.

| | |
|---|---|
| $'$ | [indicates final state of an operation] |
| ? | [indicates input to an operation] |
| ! | [indicates output of an operation] |
| $\Delta$ | [indicates the schema results in a change to the state space] |
| $\Xi$ | [indicates the schema does not result in a change to the state space] |
| $\gg$ | [indicates output of the left schema is input to the right schema] |

## 1.2 Types

Objects have a type which characterizes them and distinguish them from other kinds of objects.

- Basic types are sets of objects which have no internal structure of interest meaning the concrete definition of the members is not relevant, only their shared type.
- Free types are used to describe (potentially nested and/or recursive) sets of objects. In the most simple case, a free type can be an enumeration of constants.

Within the xAPI Formal Specification, both of these types are used to describe the Inverse Functional Identifier property.

- Introduction of the basic types $MBOX$, $MBOX\_SHA1SUM$, $OPENID$ and $ACCOUNT$ allows the specification to talk about these constraints within the xAPI specification without defining their exact structure
- The free type $IFI$ is defined as one of the above basic types meaning an object of type $IFI$ is of type $MBOX$ or $MBOX\_SHA1SUM$ or $OPENID$ or $ACCOUNT$.

Types can be composed together to form composite types and thus complex objects.

$$[MBOX, MBOX\_SHA1SUM, OPENID, ACCOUNT]$$
$$IFI ::= MBOX \mid MBOX\_SHA1SUM \mid OPENID \mid ACCOUNT$$

Within the xAPI Formal Specification, $IFI$ is used within the definition of an *agent* as presented in the schema *Agent*.

*Agent*

$$agent : AGENT$$
$$objectType : OBJECTTYPE$$
$$name : \mathbb{F}_1 \#1$$
$$ifi : IFI$$

---

$$objectType = Agent$$
$$agent = \{ifi\} \cup \mathbb{P}\{name, objectType\}$$

See section 2.2, pages 28 to 34, and chapter 3, pages 42 to 85, for more information about Schemas and the Z Language.

## 1.3 Sets

A collection of elements that all share a type. A set is characterized solely by which objects are members and which are not. Both the order and repetition of objects are ignored. Sets are written in one of two ways:

- listing their elements
- by a property which is characteristic of the elements of the set.

such that the following law from page 55 holds for some object y

$$y \in \{x_1, ..., x_n\} \iff y = x_1 \lor ... \lor y = x_n$$

## 1.4 Ordered Pairs

Two objects $(x, y)$ where $x$ is paired with $y$. An n-tuple is the pairing of n objects together such that equality between two n-tuple pairs is given by the law from page 55

$$(x_1, ..., x_n) = (y_1, ..., y_n) \iff x_1 = y_1 \land ... \land x_n = y_n$$

When ordered pairs are used with respect to application (as seen on page 60)

$$fx \Rightarrow f(x) \iff (x, y) \in f$$

which states that $f(x)$ is defined if and only if there is a unique value $y$ which result from $fx$ Additionally, application associates to the left

$$fxy \Rightarrow (fx)y \Rightarrow (f(x), y)$$

meaning $f(x)$ results in a function which is then applied to $y$.

## 1.5 Sequences

A collection of elements where their ordering matters such that

$$\langle a_1, ..., a_n \rangle \Rightarrow \{1 \mapsto a_1, ..., n \mapsto a_n\}$$

as seen on page 115. Additionally, iseq is used to describe a sequence whose members are distinct.

## 1.6 Bags

A collection of elements where the number of times an element appears in the collection is meaningful.

$$[\![a_1, ..., a_n]\!] \Rightarrow \{a_1 \mapsto k_1, ..., k_n \mapsto k_n\}$$

As described on page 124, each element $a_i$ appears $k_i$ times in the list $a_1, ..., a_n$ such that the number of occurrences of $a_i$ within bag $A$ is returned by

$$count\, A\, a_i \ \equiv A \# a_i$$

## 1.7 Maps

This document introduces a named subcategory of sets, *map* of the free type $KV$, which are akin to sequences and bags. To enumerate the members of a *map*, $\langle\!\langle ... \rangle\!\rangle$ is used but should not be confused with $d_i \langle\!\langle E_i[T] \rangle\!\rangle$ within a Free Type definition. The distinction between the two usages is context dependent but in general, if $\langle\!\langle ... \rangle\!\rangle$ is used outside of a constructor declaration within a Free Type definition, it should be assumed to represent a *map*.

$$KV ::= base \,|\, associate \langle\!\langle KV \times X \times Y \rangle\!\rangle$$

where

| | |
|---|---|
| *base* | [is a constant which is the empty $KV \Rightarrow \langle\!\langle \rangle\!\rangle$] |
| *associate* | [is a constructor and is inferred to be an injection] |

The full enumeration of all properties, constraints and functions specific to a *map* with type $KV$ will be defined elsewhere but *associate* can be understood to (in the most basic case) operate as follows.

$$associate(base, x_i, y_i) = \langle\!\langle (x_i, y_i) \rangle\!\rangle \Rightarrow \langle\!\langle x_i \mapsto y_i \rangle\!\rangle$$

The enumeration of a *map* was chosen to be $\langle\!\langle ... \rangle\!\rangle$ as a *map* is a collection of injections such that if $M$ is the result of $associate(base, x_i, y_i)$ from above then

$$atKey(M, x_i) = y_i \iff x_i \mapsto y_i \wedge (x_i, y_i) \in M$$

## 1.8 Select Operations and Symbols

The follow are defined in Chapter 4 (The Mathematical Tool-kit) within the reference manual and are used extensively throughout this document. In many cases, the functions listed here will serve as Operations in the context of Primitives and Algorithms.

### 1.8.1 Functions

$\pfun$ [relate each x ∈ X to at most one y ∈ Y, page 105]

$\fun$ [relate each x ∈ X to exactly one y ∈ Y, page 105]

$\pinj$ [map different elements of x to different y, page 105]

$\inj$ [$\pinj$ that are also $\fun$, page 105]

$\psurj$ [$X \pfun Y$ where whole of Y is the range, page 105]

$\surj$ [$X \pfun Y$ whole of X as domain and whole of Y as range, page 105]

$\bij$ [map x ∈ X one-to-one with y ∈ Y, page 105]

$$X \pfun Y == \{ f : X \rel Y \mid (\forall x : X; y1, y2 : Y \bullet (x \mapsto y_1 \in f \ \land \ (x \mapsto y_2) \in f \Rightarrow y_1 = y_2))\}$$
$$X \fun Y == \{ f : X \pfun Y \mid \dom f = X\}$$
$$X \pinj Y == \{ f : X \pfun Y \mid (\forall x_1, x_2 : \dom f \bullet f(x_1) = f(x_2) \Rightarrow x_1 = x_2)\}$$
$$X \inj Y == (X \pinj Y) \cap (X \fun Y)$$
$$X \psurj Y == \{ f : X \pinj Y \mid \ran f = Y\}$$
$$X \surj Y == (X \psurj Y) \cap (X \fun Y)$$
$$X \bij Y == (X \surj Y) \cap (X \inj Y)$$

### 1.8.2 Ordered Pairs, Maplet and Composition of Relations

$first$ [returns the first element of an ordered pair, page 93]

$second$ [returns the second element of an ordered pair, page 93]

$\mapsto$ [maplet is a graphic way of expressing an ordered pair, page 95]

dom [set of all x ∈ X related to at least one y ∈ Y by R, page 96]

ran [set of all y ∈ Y related to at least one x ∈ X by R, page 96]

$\fatsemi$ [The composition of two relationships, page 97]

$\circ$ [The backward composition of two relationships, page 97]

$[X, Y]$

$$first : X \times Y \fun X$$
$$second : X \times Y \fun Y$$

---

$$\forall x : X; y : Y \bullet first(x, y) = x \land second(x, y) = y$$

$[X, Y]$

$$\_ \mapsto \_ : X \times Y \to X \times Y$$

---

$$\forall x : X; y : Y \bullet$$
$$x \mapsto y = (x, y)$$

$[X, Y]$

$$\mathrm{dom} : (X \leftrightarrow Y) \to \mathbb{P}\, X$$
$$\mathrm{ran} : (X \leftrightarrow Y) \to \mathbb{P}\, Y$$

---

$$\forall R : X \leftrightarrow Y \bullet$$
$$\mathrm{dom}\, R = \{\, x : X;\ y : Y \mid x \underline{R} y \bullet x \} \land$$
$$\mathrm{ran}\, R = \{\, x : X;\ y : Y \mid x \underline{R} y \bullet y \}$$

$[X, Y, Z]$

$$\_ \fatsemi \_ : (X \leftrightarrow Y) \times (Y \leftrightarrow Z) \to (X \leftrightarrow Z)$$
$$\_ \circ \_ : (Y \leftrightarrow X) \times (X \leftrightarrow Y) \to (X \leftrightarrow X)$$

---

$$\forall Q : X \leftrightarrow Y; R : Y \leftrightarrow Z \bullet$$
$$Q \fatsemi R = R \circ Q = \{\, x : X; y : Y; z : Z \mid$$
$$x \underline{Q} y \ \land\ y \underline{R} z \bullet x \mapsto z \}$$

### 1.8.3 Numeric

| | |
|---|---|
| *succ* | [the next natural number, page 109] |
| .. | [set of integers within a range, page 109] |
| # | [number of members of a set, page 111] |
| min | [smallest number in a set of numbers, page 113] |
| max | [largest number in a set of numbers, page 113] |

$$succ : \mathbb{N} \to \mathbb{N}$$
$$\_ .. \_ : \mathbb{Z} \times \mathbb{Z} \to \mathbb{P}\, \mathbb{Z}$$

---

$$\forall n : \mathbb{N} \bullet succ(n) = n + 1$$
$$foralla, b : \mathbb{Z} \bullet$$
$$a .. b = \{\, k : \mathbb{Z} \mid a \leq k \leq b \}$$

$[X]$

$$\# : \mathbb{F}\, X \to \mathbb{N}$$

---

$$\forall S : \mathbb{F}\, X \bullet$$
$$\#S = (\mu\, n : \mathbb{N} \mid (\exists f : 1 .. n \rightarrowtail X \bullet \mathrm{ran}\, f = S))$$

$$
\begin{array}{|l}
\min : \mathbb{P}_1\, \mathbb{Z} \pfun \mathbb{Z} \\
\max : \mathbb{P}_1\, \mathbb{Z} \pfun \mathbb{Z} \\
\hline
\min = \{\, S : \mathbb{P}_1\, \mathbb{Z}; m : \mathbb{Z} \mid \\
\qquad m \in S \land (\forall n : S \bullet m \leq n) \bullet S \mapsto m\} \\
\max = \{\, S : \mathbb{P}_1\, \mathbb{Z}; m : \mathbb{Z} \mid \\
\qquad m \in S \land (\forall n : S \bullet m \geq n) \bullet S \mapsto m\}
\end{array}
$$

### 1.8.4 Sequences



$$
\begin{array}{|l}
[X] \\
\hline
\_ \frown \_ : \seq X \times \seq X \to \seq X \\
rev : \seq X \to \seq X \\
\hline
\forall s, t : \seq X \bullet \\
\qquad s \frown t = s \cup \{\, n : \dom t \bullet n + \# s \mapsto t(n)\} \\
\forall s : \seq X \bullet \\
\qquad rev s = (\lambda\, n : \dom s \bullet s(\# s - n + 1)) \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
[X] \\
\hline
head, last : \seq_1 X \to X \\
tail, front : \seq_1 X \to \seq X \\
\hline
\forall s : \seq_1 X \bullet \\
\qquad head\, s = s(1) \land \\
\qquad last\, s = s(\# s) \land \\
\qquad tail\, s = (\lambda\, n : 1 \upto \# s - 1 \bullet s(n + 1)) \land \\
\qquad front\, s = (1 \upto \# s - 1) \ndres s \\
\hline
\end{array}
$$

$[X]$

$$\_ \upharpoonright \_ : \mathbb{P}\,\mathbb{N}_1 \times \operatorname{seq} X \to \operatorname{seq} X$$
$$\_ \uparrow \_ : \operatorname{seq} X \times \mathbb{P}\, X \to \operatorname{seq} X$$
$$squash : (\mathbb{N}_1 \pfun X) \to \operatorname{seq} X$$

---

$$\forall U : \mathbb{P}\,\mathbb{N}_1; s : \operatorname{seq} X \bullet U \upharpoonright s = squash(U \lhd s)$$
$$\forall s : \operatorname{seq} X; V : \mathbb{P}\, X \bullet s \uparrow V = squash(s \rhd V)$$
$$\forall f : \mathbb{N}_1 \pfun X \bullet squash f = f \circ (\mu\, p : 1 \upto \# f \pinj \operatorname{dom} f \mid p \circ succ \circ p^\sim \subseteq (\_ < \_))$$

$[X]$

$$\frown / : \operatorname{seq}(\operatorname{seq} X) \to \operatorname{seq} X$$

---

$$\frown / \langle\rangle = \langle\rangle$$
$$\forall s : \operatorname{seq} X \bullet \frown / \langle s \rangle = s$$
$$\forall q, r : \operatorname{seq}(\operatorname{seq} X) \bullet \frown / (q \frown r) = (\frown / q) \frown (\frown / r)$$

$[I, X]$

$$\mathsf{disjoint}\ \_ : \mathbb{P}(I \pfun \mathbb{P}\, X)$$
$$\_\ \mathsf{partition}\ \_ : (I \pfun \mathbb{P}\, X) \rel \mathbb{P}\, X$$

---

$$\forall S : I \pfun \mathbb{P}\, X; T : \mathbb{P}\, X \bullet$$
$$(\mathsf{disjoint}\ S \iff (\forall i, j : \operatorname{dom} S \mid i \neq j \bullet S(i) \cap S(j) = \emptyset)) \land$$
$$(S\ \mathsf{partition}\ T \iff \mathsf{disjoint}\ S \land \bigcup \{ i : \operatorname{dom} S \bullet S(i) \} = T)$$

### 1.8.5 Bags

*count*, # [the number of times something appears in a bag, page 124]

$\otimes$ [scaling across a bag, page 124]

$\uplus$ [union of two bags, sum of occurrences, page 126]

$\biguplus$ [bag difference, subtract occurrences or zero if negative, page 126]

*items* [conversion from seq to bag, page 127]

$[X]$

$$
\begin{array}{l}
count : \mathrm{bag}\, X \rightarrowtail (X \to \mathbb{N}) \\
\_ \# \_ : \mathrm{bag}\, X \times X \to \mathbb{N} \\
\_ \otimes \_ : \mathbb{N} \times \mathrm{bag}\, X \to \mathrm{bag}\, X \\
\hline
\forall B : \mathrm{bag}\, X \bullet \\
\quad count B = (\lambda x : X \bullet 0) \oplus B \\
\forall x : X; B : \mathrm{bag}\, x \bullet \\
\quad B \# x = count\, B\, x \\
\forall n : \mathbb{N}; B : \mathrm{bag}\, X; x : X \bullet \\
\quad (n \otimes B) \# x = n * (B \# x)
\end{array}
$$

$[X]$

$$
\begin{array}{l}
\_ \uplus \_ , \_ \cup \_ : \mathrm{bag}\, X \times \mathrm{bag}\, X \to \mathrm{bag}\, X \\
\hline
\forall B\, , C : \mathrm{bag}\, X; x : X \bullet \\
\quad (B \uplus C) \# x = B \# x + C \# x \wedge \\
\quad (B \cup C) \# x = \max\{B \# x - C \# x, 0\}
\end{array}
$$

$[X]$

$$
\begin{array}{l}
items : \mathrm{seq}\, X \to \mathrm{bag}\, X \\
\hline
\forall s : \mathrm{seq}\, X; x : X \bullet \\
\quad (items\, s) \# x = \#\{ i : \mathrm{dom}\, s \mid s(i) = x \}
\end{array}
$$