User - Defined Squash Function

The squash operator appears in the Z notation glossary but is not built into the fuzz type-checker. However, we can define it ourselves as a user-defined function using an axiomatic specification.

Squash takes a partial function from N1 (positive naturals) to some type X and compacts it into a sequence by removing gaps in the domain. For example, $\{1 \mapsto a, 3 \mapsto b, 7 \mapsto c\}$ becomes $\langle a, b, c \rangle$.

Example 1 : Axiomatic Definition of Squash

We define squash axiomatically by specifying its key properties using only fuzz-compatible notation.

$$[X]$$

$$\mathit{squashFunc} : (\mathbb{N}_1 \pfun X) \longrightarrow \mathrm{seq}\ X$$

$$\forall f : \mathbb{N}_1 \pfun X \bullet \mathrm{ran}(\mathit{squashFunc}(f)) = \mathrm{ran}\, f \land \#(\mathit{squashFunc}(f)) = \#(\mathrm{ran}\, f)$$

This axiomatically defines squashFunc (a user-defined version of squash) with two key properties that hold for all partial functions f from N1 to X: (1) squashFunc preserves the *range* : *ran* (squashFunc f) = ran f, and (2) the resulting sequence length equals the range cardinality: # (squashFunc f) = # (ran f). These properties characterize squashFunc as compacting partial functions by removing domain gaps while preserving all elements and their relative order.

Example 2 : Using Squash on a Simple Function

Demonstrate squash on a sparse partial function.

$$f1 : \mathbb{N}_1 \pfun \mathbb{N}$$
$$s1 : \mathrm{seq}\ \mathbb{N}$$

$$f1 = \{1 \mapsto 10, 3 \mapsto 30, 5 \mapsto 50\} \land s1 = \mathit{squashFunc}(f1)$$

The result s1 is $\langle 10, 30, 50 \rangle$, containing exactly the range elements in the order of their original indices.

Example 3 : Squash with Letters

$$[\mathit{Letter}]$$

$$a : \mathit{Letter}$$
$$b : \mathit{Letter}$$
$$c : \mathit{Letter}$$
$$d : \mathit{Letter}$$
$$f2 : \mathbb{N}_1 \pfun \mathit{Letter}$$
$$s2 : \mathrm{seq}\ \mathit{Letter}$$

$$f2 = \{2 \mapsto a, 4 \mapsto b, 7 \mapsto c, 9 \mapsto d\} \land s2 = \mathit{squashFunc}(f2)$$

The result s2 is $\langle a, b, c, d \rangle$. The gaps at positions 1, 3, 5, 6, 8 are removed.

Example 4 : Empty Function

$$
\begin{array}{|l}
f3 : \mathbb{N}_1 \pfun \mathbb{N} \\
s3 : \operatorname{seq} \mathbb{N} \\
\hline
f3 = \{\} \land s3 = \mathit{squashFunc}(f3)
\end{array}
$$

Squashing an empty function yields an empty *sequence* : $s3 = \langle\rangle$.

Example 5 : Continuous Function (No Gaps)

$$
\begin{array}{|l}
f4 : \mathbb{N}_1 \pfun \mathbb{N} \\
s4 : \operatorname{seq} \mathbb{N} \\
\hline
f4 = \{1 \mapsto 100, 2 \mapsto 200, 3 \mapsto 300\} \land s4 = \mathit{squashFunc}(f4)
\end{array}
$$

When the function has no gaps ($\operatorname{dom} f4 = 1..3$), squashFunc simply converts it to a *sequence* : $s4 = \langle 100, 200, 300 \rangle$.

Example 6 : Relationship Between Function land Sequence

For any partial function f with a continuous domain starting at 1, squashFunc is the identity.

$$
\begin{array}{|l}
\mathit{continuous\_N} : \mathbb{N}_1 \pfun \mathbb{N} \\
\mathit{s\_continuous\_N} : \operatorname{seq} \mathbb{N} \\
\hline
\operatorname{dom} \mathit{continuous\_N} = 1 \mathinner{..} \#(\operatorname{ran} \mathit{continuous\_N}) \land \mathit{s\_continuous\_N} = \mathit{squashFunc}(\mathit{continuous\_N})
\end{array}
$$

In this case, continuous_N is already a sequence, and squashFunc $\mathit{continuous\_N} = \mathit{continuous\_N}$ as a sequence.

Example 7 : Squash Preserves Injectivity

If the input function is injective, the output sequence has no duplicates.

$$
\begin{array}{|l}
\mathit{inj\_func} : \mathbb{N}_1 \pfun \mathbb{N} \\
\mathit{inj\_seq} : \operatorname{seq} \mathbb{N} \\
\hline
(\forall i, j : \operatorname{dom} \mathit{inj\_func} \bullet \mathit{inj\_func}(i) = \mathit{inj\_func}(j) \Rightarrow i = j) \land \mathit{inj\_seq} = \mathit{squashFunc}(\mathit{inj\_func})
\end{array}
$$

Since inj_func is injective (one-to-one), inj_seq contains no duplicate elements.

Example 8 : Practical Use Case - Video Database

$$[\mathit{VideoID}, \mathit{Title}]$$

$$
\begin{array}{|l}
\mathit{sparse\_catalog} : \mathbb{N}_1 \pfun (\mathit{VideoID} \times \mathit{Title}) \\
\mathit{compact\_catalog} : \operatorname{seq} (\mathit{VideoID} \times \mathit{Title}) \\
\hline
\mathit{compact\_catalog} = \mathit{squashFunc}(\mathit{sparse\_catalog})
\end{array}
$$

A sparse catalog with gaps (deleted entries) can be compacted into a continuous sequence while preserving the order of remaining entries.