1 Semantica operazionale

1.1 Set

$$\text{Set}(type, [v_1, ..., v_n]) = \{v_i \mid v_i : type, \forall i, j \in [1, n] \text{ and } i \neq j \Rightarrow v_i \neq v_j\}$$

1.2 EmptySet

$$\frac{type \in types}{\text{env} \triangleright \text{EmptySet}(type) \Rightarrow \text{Set}(type, [])}$$

1.3 Singleton

$$\frac{\text{env} \triangleright exp \Rightarrow val, val : type}{\text{env} \triangleright \text{Singleton}(exp, type) \Rightarrow \text{Set}(type, [val])}$$

1.4 Insert

$$\frac{\text{env} \triangleright e_1 \Rightarrow v_1, e_2 \Rightarrow v_2, v_1 : SetType(type), v_2 : type, v_2 \notin v_1}{\text{env} \triangleright \text{Insert}(e_1, e_2) \Rightarrow v_1 \cup \{v_2\} : SetType(type)}$$

1.5 Remove

$$\frac{\text{env} \triangleright e_1 \Rightarrow v_1, e_2 \Rightarrow v_2, v_1 : SetType(type), v_2 : type, \mid v_1 \mid \neq 0}{\text{env} \triangleright \text{Remove}(e_1, e_2) \Rightarrow v_1 \setminus \{v_2\} : SetType(type)}$$

1.6 IsEmpty

$$\frac{\text{env} \triangleright e \Rightarrow v, v : SetType(type), \mid v \mid = 0}{\text{env} \triangleright \text{IsEmpty}(e) \Rightarrow true}$$

$$\frac{\text{env} \triangleright e \Rightarrow v, v : SetType(type), \mid v \mid \neq 0}{\text{env} \triangleright \text{IsEmpty}(e) \Rightarrow false}$$

1.7 findMax

$$\frac{\text{env} \triangleright e \Rightarrow v, v : SetType(type), \mid v \mid \neq 0, v_{max} \in v, \forall v_i \in v : v_{max} \geq v_i}{\text{env} \triangleright \text{IsEmpty}(e) \Rightarrow v_{max}}$$

1.8 findMin

$$\frac{\text{env} \triangleright e \Rightarrow v, v : SetType(type), \mid v \mid \neq 0, v_{min} \in v, \forall v_i \in v : v_{min} \leq v_i}{\text{env} \triangleright \text{IsEmpty}(e) \Rightarrow v_{min}}$$

1.9 isIn

$$\frac{\text{env} \triangleright e_1 \Rightarrow v_1, e_2 \Rightarrow v_2, v_1 : \mathit{SetType}(\mathit{type}), v_2 : \mathit{type}, v_2 \in v_1}{\text{env} \triangleright \text{IsIn}(e_1, e_2) \Rightarrow \mathit{true}}$$

$$\frac{\text{env} \triangleright e_1 \Rightarrow v_1, e_2 \Rightarrow v_2, v_1 : \mathit{SetType}(\mathit{type}), v_2 : \mathit{type}, v_2 \notin v_1}{\text{env} \triangleright \text{IsIn}(e_1, e_2) \Rightarrow false}$$

1.10 isSubsetOf

$$\frac{\text{env} \triangleright e_1 \Rightarrow v_1, e_2 \Rightarrow v_2, v_1 : \mathit{SetType}(\mathit{type}), v_2 : \mathit{SetType}(\mathit{type}), v_2 \subseteq v_1}{\text{env} \triangleright \text{isSubsetOf}(e_1, e_2) \Rightarrow \mathit{true}}$$

$$\frac{\text{env} \triangleright e_1 \Rightarrow v_1, e_2 \Rightarrow v_2, v_1 : \mathit{SetType}(\mathit{type}), v_2 : \mathit{SetType}(\mathit{type}), v_2 \not\subseteq v_1}{\text{env} \triangleright \text{isSubsetOf}(e_1, e_2) \Rightarrow false}$$

1.11 Union

$$\frac{\text{env} \triangleright e_1 \Rightarrow v_1, e_2 \Rightarrow v_2, v_1 : \mathit{SetType}(\mathit{type}), v_2 : \mathit{SetType}(\mathit{type})}{\text{env} \triangleright \text{Union}(e_1, e_2) \Rightarrow v_1 \cup v_2}$$

1.12 Intersect

$$\frac{\text{env} \triangleright e_1 \Rightarrow v_1, e_2 \Rightarrow v_2, v_1 : \mathit{SetType}(\mathit{type}), v_2 : \mathit{SetType}(\mathit{type})}{\text{env} \triangleright \text{Intersect}(e_1, e_2) \Rightarrow v_1 \cap v_2}$$

1.13 Difference

$$\frac{\text{env} \triangleright e_1 \Rightarrow v_1, e_2 \Rightarrow v_2, v_1 : \mathit{SetType}(\mathit{type}), v_2 : \mathit{SetType}(\mathit{type})}{\text{env} \triangleright \text{Difference}(e_1, e_2) \Rightarrow v_1 \setminus v_2}$$

1.14 ForAll

$$\frac{\begin{array}{c}\text{env} \triangleright e_1 \Rightarrow fun, e_2 \Rightarrow v, fun : \mathit{Closure}(argIde, funExp, fEnv), v : \mathit{SetType} \\ \forall v_i \in v \ \text{fEnv}[v_i/argIde] \triangleright funExp \Rightarrow \mathit{true}\end{array}}{\text{env} \triangleright \text{ForAll}(e_1, e_2) \Rightarrow \mathit{true}}$$

$$\frac{\begin{array}{c}\text{env} \triangleright e_1 \Rightarrow fun, e_2 \Rightarrow v, fun : \mathit{Closure}(argIde, funExp, fEnv), v : \mathit{SetType} \\ \exists v_i \in v \ \text{fEnv}[v_i/argIde] \triangleright funExp \Rightarrow false\end{array}}{\text{env} \triangleright \text{ForAll}(e_1, e_2) \Rightarrow false}$$

1.15 Exists

$$\frac{\begin{array}{c}\text{env} \triangleright e_1 \Rightarrow fun, e_2 \Rightarrow v, fun : Closure(argIde, funExp, fEnv), v : SetType \\ \exists v_i \in v \; \text{fEnv}[v_i/argIde] \triangleright funExp \Rightarrow true\end{array}}{\text{env} \triangleright \text{Exists}(e_1, e_2) \Rightarrow true}$$

$$\frac{\begin{array}{c}\text{env} \triangleright e_1 \Rightarrow fun, e_2 \Rightarrow v, fun : Closure(argIde, funExp, fEnv), v : SetType \\ \forall v_i \in v \; \text{fEnv}[v_i/argIde] \triangleright funExp \Rightarrow false\end{array}}{\text{env} \triangleright \text{Exists}(e_1, e_2) \Rightarrow false}$$

1.16 Filter

$$\frac{\begin{array}{c}\text{env} \triangleright e_1 \Rightarrow fun, e_2 \Rightarrow v, fun : Closure(argIde, funExp, fEnv), v : SetType(type) \\ filteredSet = \{v_i \mid v_i \in v \text{ and } \text{fEnv}[v_i/argIde] \triangleright funExp \Rightarrow true\}\end{array}}{\text{env} \triangleright \text{Filter}(e_1, e_2) \Rightarrow filteredSet}$$

1.17 Map

$$\frac{\begin{array}{c}\text{env} \triangleright e_1 \Rightarrow fun, e_2 \Rightarrow v, fun : Closure(argIde, funExp, fEnv), v : SetType(type) \\ mappedSet = \{m_i \mid \forall v_i \in v \; \text{fEnv}[v_i/argIde] \triangleright funExp \Rightarrow m_i\}\end{array}}{\text{env} \triangleright \text{Map}(e_1, e_2) \Rightarrow mappedSet}$$