Semantica operazionale

Eset

$$\frac{(\forall\ exp \in expList\ .\ env\ \triangleright exp\ \Rightarrow evt\ \wedge lang\_typecheck\ (type,\ evt)),\quad \{evt \mid (\exists\ expr \in expList\ .\ env\ \triangleright\ exp \Rightarrow\ evt)\} \Rightarrow evtList}{env\ \triangleright Eset(type, expList) \Rightarrow Set(type, evtList)}$$

Singleton

$$\frac{\{evt \mid env\ \triangleright expr\ \Rightarrow evt\ \wedge lang\_typecheck\ (type,\ evt)\} \Rightarrow evtList}{env\ \triangleright Singleton(type, exp) \Rightarrow Set(type, evtList)}$$

Of

$$\frac{(\forall\ exp \in expList\ .\ env\ \triangleright exp\ \Rightarrow evt\ \wedge lang\_typecheck\ (type,\ evt)),\quad \{evt \mid (\exists\ expr \in expList\ .\ env\ \triangleright\ exp \Rightarrow\ evt)\} \Rightarrow evtList}{env\ \triangleright Eset(type, expList) \Rightarrow Set(type, evtList)}$$

EmptySet

$$\frac{-}{env\ \triangleright EmptySet(type) \Rightarrow Set(type, \emptyset)}$$

Union

$$\frac{(env\ \triangleright set_a \Rightarrow Set(type_a, list_a)),\ (env\ \triangleright set_b \Rightarrow Set(type_b, list_a)),\ type_a = type_b))}{env\ \triangleright Union(set_a, set_b) \Rightarrow Set(type_a, (list_a \cup list_b))}$$

DiffSet

$$\frac{(env\ \triangleright set_a \Rightarrow Set(type_a, list_a)),\ (env\ \triangleright set_b \Rightarrow Set(type_b, list_a)),\ (type_a = type_b))}{env\ \triangleright Union(set_a, set_b) \Rightarrow Set(type_a, list_a \setminus list_b)}$$

Intersection

$$\frac{(env\ \triangleright set_a \Rightarrow Set(type_a, list_a)),\ (env\ \triangleright set_b \Rightarrow Set(type_b, list_a)),\ (type_a = type_b))}{env\ \triangleright Union(set_a, set_b) \Rightarrow Set(type_a, list_a \cap list_b)}$$

Insert

$$\frac{(env\ \triangleright set_a \Rightarrow Set(type_a, list_a)),\ (env\ \triangleright expr \Rightarrow value),\ (lang\_typecheck(type_a,\ value))}{env\ \triangleright Insert(set_a, exp) \Rightarrow Set(type_a, list_a \cup \{value\})}$$

Remove

$$\frac{(env \triangleright set_a \Rightarrow Set(type_a, list_a)),\ (env \triangleright expr \Rightarrow value),\ (lang\_typecheck(type_a,\ value))}{env \triangleright Insert(set_a, exp) \Rightarrow Set(type_a, list_a \setminus \{value\})}$$

Contains

$$\frac{(env \triangleright set_a \Rightarrow Set(type_a, list_a)),\ (env \triangleright expr \Rightarrow value)}{env \triangleright Contains(set_a, exp) \Rightarrow Bool(value \in list_a)}$$

IsEmpty

$$\frac{env \triangleright set_a \Rightarrow Set(type_a, list_a)}{env \triangleright IsEmpty(set_a, exp) \Rightarrow Bool(list_a = \emptyset)}$$

Subset

$$\frac{(env \triangleright set_a \Rightarrow Set(type_a, list_a)),\ (env \triangleright set_b \Rightarrow Set(type_b, list_b)),\ (type_a = type_b)}{env \triangleright Subset(set_a, set_b) \Rightarrow Bool(set_a \subseteq set_b)}$$

Min

$$\frac{(env \triangleright set_a \Rightarrow Set(type_a, list_a)),\ \min\{list_a\} \Rightarrow minValue}{env \triangleright Min(set_a) \Rightarrow Int(minValue)}$$

Max

$$\frac{(env \triangleright set_a \Rightarrow Set(type_a, list_a)),\ \max\{list_a\} \Rightarrow maxValue}{env \triangleright Max(set_a) \Rightarrow Int(maxValue)}$$

ForAll

$$\frac{\begin{array}{l}(env \triangleright set_a \Rightarrow Set(type_a, list_a)),\ (env \triangleright func\ \Rightarrow FunVal(ide, expr, arg)),\\ set_a \Rightarrow (type, setList), (\forall exp \in setList\ .\ env \triangleright FunCall(FunVal(ide, expr, exp)) \Rightarrow Bool(true)) \Rightarrow ret\end{array}}{env \triangleright\ ForAll(set_a, func) \Rightarrow Bool(ret)}$$

Exists

$$\frac{\begin{array}{l}(env \triangleright set_a \Rightarrow Set(type_a, list_a)),\ (env \triangleright func\ \Rightarrow FunVal(ide, expr, arg)),\\ set_a \Rightarrow (type, setList), (\exists exp \in setList\ .\ env \triangleright FunCall(FunVal(ide, expr, exp)) \Rightarrow Bool(true)) \Rightarrow ret)\end{array}}{env \triangleright\ ForAll(set_a, func) \Rightarrow Bool(ret)}$$

Filter

$$\frac{(env \triangleright set_a \Rightarrow Set(type_a, list_a)),\ (env \triangleright func \Rightarrow FunVal(ide, expr, arg)),\ set_a \Rightarrow (type, setList), \{expr \in list_a \mid env \triangleright FunCall(FunVal(ide, expr, exp)) \Rightarrow Bool(true)\} \Rightarrow list_b}{env \triangleright Filter(set_a, func) \Rightarrow Set(type_a, list_b)}$$

Map

$$\frac{\begin{array}{l}(env \triangleright set_a \Rightarrow Set(type_a, list_a)),\ (env \triangleright func \Rightarrow FunVal(ide, expr, arg)),\\ set_a \Rightarrow (type, setList),\ env \triangleright IsEmpty(set_a) \Rightarrow empty,\\ \neg empty \implies get\_lang\_type(env \triangleright FunCall(FunVal(ide, expr, arg))) \Rightarrow type_b,\\ empty \implies type_b, \{evt \mid env \triangleright FunCall(FunVal(ide, expr, arg)) = evt\} \Rightarrow list_b\end{array}}{env \triangleright Map(set_a, func) \Rightarrow Set(type_b, list_b)}$$