# PROGETTO DI PROGRAMMAZIONE 2

**UNIVERSITÁ DI PISA**
**DIPARTIMENTO IN INFORMATICA**
**LAUREA IN INFORMATICA**

ADRIEN KOUMGANG TEGANTCHOUANG
MATRICOLA 582092

15 dicembre 2020

## Indice

# 1 Regole operazionali

## 1.1 EmptySet

$$\frac{type \in \{"int","float","string"\}}{env \triangleright EmptySet(type) \Rightarrow setT(type, [])}$$

## 1.2 Singleton

$$\frac{\frac{type \in \{"int","float","string"\}}{env \triangleright value \Rightarrow evalue \quad typeof(evalue) = type}}{env \triangleright Singleton(value, type) \Rightarrow setT(type, evalue)}$$

## 1.3 Of

$$\frac{\frac{\frac{type \in \{"int","float","string"\}}{env \triangleright values \Rightarrow evalues}}{\forall v \in evalues \,.\, typeof(v) = type}}{env \triangleright Of(type, values) \Rightarrow setT(type, evalues)}$$

## 1.4 Union

$$\frac{\frac{\frac{env \triangleright set1 \Rightarrow setT(type1, evalues1)}{env \triangleright set2 \Rightarrow setT(type1, evalues2)}}{\frac{type = type1 = type2}{values = evalues1 \cup evalues2}}}{env \triangleright Union(set1, set2) \Rightarrow setT(type, values)}$$

## 1.5 Inter

$$\frac{\frac{\frac{env \triangleright set1 \Rightarrow setT(type1, evalues1)}{env \triangleright set2 \Rightarrow setT(type1, evalues2)}}{\frac{type = type1 = type2}{values = evalues1 \cap evalues2}}}{env \triangleright Union(set1, set2) \Rightarrow setT(type, values)}$$

## 1.6 Diff

$$\frac{\frac{\frac{env \triangleright set1 \Rightarrow setT(type1, evalues1)}{env \triangleright set2 \Rightarrow setT(type1, evalues2)}}{\frac{type = type1 = type2}{values = evalues1 - evalues2}}}{env \triangleright Diff(set1, set2) \Rightarrow setT(type, values)}$$

## 1.7 Insert

$$\frac{\frac{\frac{env \triangleright set \Rightarrow setT(type, evalues)}{env \triangleright element \Rightarrow E}}{\frac{type = typeof(E)}{finalValues = evalues \cup \{E\}}}}{env \triangleright Union(set, element) \Rightarrow setT(type, finalValues)}$$

## 1.8 Remove

$$\frac{\frac{\frac{env \triangleright set \Rightarrow setT(type, values)}{env \triangleright element \Rightarrow E}}{\frac{type = typeof(E)}{finalValues = evalues - \{E\}}}}{env \triangleright Union(set, element) \Rightarrow setT(type, finalValues)}$$

## 1.9 IsEmpty

$$\frac{\frac{env \triangleright set \; setT(type, values)}{if \; values = [] \; then \; b = Bool(true)}}{else \; b = Bool(false)}$$
$$env \triangleright IsEmpty(set) \Rightarrow b$$

## 1.10 HasElement

$$\frac{\frac{\frac{env \triangleright set \Rightarrow setT(type, evalues)}{env \triangleright element \Rightarrow E}}{type = typeof(E)}}{if \; \exists \; v \in evalues \mid v = E \; then \; b = Bool(true) \; else \; b = Boolf(false)}$$
$$env \triangleright HasElement(set, element) \Rightarrow b$$

## 1.11 IsSubSet

$$\frac{\frac{\frac{env \triangleright set1 \Rightarrow setT(type1, evalues1)}{env \triangleright set2 \Rightarrow setT(type1, evalues2)}}{if \; \forall \; v \in evalues \quad \exists \; w \in evales2 \mid v = w \; then \; b = Bool(true)}}{else \; b = Bool(false)}$$
$$env \triangleright IsSubSet(set1, set2) \Rightarrow b$$

## 1.12 MaxSet

$$\frac{\frac{env \triangleright set \Rightarrow SetT(type, evalues)}{type \in \{"int" \; "float" \; "string"\}}}{maxE = v \in evalues \mid \forall \; w \in evalues \; v \geq w}$$
$$env \triangleright MaxSet(set) \Rightarrow maxE$$

## 1.13 MinSet

$$\frac{\frac{env \triangleright set \Rightarrow SetT(type, evalues)}{type \in \{"int" \; "float" \; "string"\}}}{minE = v \in evalues \mid \forall \; w \in evalues \; v \leq w}$$
$$env \triangleright MinSet(set) \Rightarrow minE$$

## 1.14 ForAll

$$\frac{\frac{\frac{\frac{env \triangleright set \Rightarrow setT(type, evalues)}{env \triangleright predicate \Rightarrow Closure(arg, ebody, s)}}{\forall \; v_i \in evalues \; env \triangleright Apply(predicate, v) \Rightarrow b_i}}{\forall \; b_i, b_j \mid i \neq j : b = b_i \; and \; b_j}}{env \triangleright ForAll(predicate, set) \Rightarrow b}$$

## 1.15 Exists

$$\frac{\frac{\frac{\frac{env \triangleright set \Rightarrow setT(type, evalues)}{env \triangleright predicate \Rightarrow Closure(arg, ebody, s)}}{\forall \; v_i \in evalues \; env \triangleright Apply(predicate, v_i) \Rightarrow b_i}}{\forall \; b_i, b_j \mid i \neq j : b = b_i \; or \; b_j}}{env \triangleright Exists(predicate, set) \Rightarrow b}$$

## 1.16 Filter

$$\frac{\frac{\frac{\frac{env \triangleright set \Rightarrow setT(type, evalues)}{env \triangleright predicate \Rightarrow Closure(arg, ebody, s)}}{\forall \; v_i \in evalues \; env \triangleright Apply(predicate, v_i) \Rightarrow b_i}}{\forall \; b_i, if \; b_i = true : resultE = resultE \cup \{v_i\}}}{env \triangleright Filter(predicate, set) \Rightarrow SetT(type, resultE)}$$

## 1.17 Map

$$\frac{\dfrac{\dfrac{env \triangleright set \Rightarrow setT(type, evalues)}{env \triangleright function \Rightarrow Closure(arg, ebody, s) \text{ or } RecClosure(f, arg, fBody, s)} \quad \dfrac{\forall\, v_i \in evalues\ env \triangleright Apply(function, v_i) \Rightarrow b_i}{\forall\, v_i : resultE = resultE \cup \{v_i\} \qquad typeResult = typeof(v_i)}}{}}{env \triangleright Map(function, set) \Rightarrow SetT(typeResult, resultE)}$$