Regole operazionali

SetEmpty of type_elts

```
env ▹ t ⟹ ("int" ∨ "string" ∨ "bool" ∨
            "fun ((targ) -> (tres))" ∨
            "recfun ((targ) -> (tres))")
-------------------------------------------
env ▹ SetEmpty(t) ⟹ Set(t, List_val [])
```

SetSingleton of type_elts * exp

```
env ▹ t ⟹ ("int" ∨ "string" ∨ "bool" ∨
            "fun ((targ) -> (tres))" ∨
            "recfun ((targ) -> (tres))")
env ▹ e ⟹ v
-------------------------------------------
env ▹ SetSingleton(t, e) ⟹ Set(t, List_val [v])
```

SetOf of type_elts * exp

```
env ▹ t ⟹ ("int" ∨ "string" ∨ "bool" ∨
            "fun ((targ) -> (tres))" ∨
            "recfun ((targ) -> (tres))")
env ▹ l ⟹ List_val [e1, e2, ..., en]
env ▹ e1, ..., en ⟹ v1, ..., vn
-------------------------------------------
env ▹ SetOf(t, l) ⟹ Set(t, List_val [v1, ..., vn])
```

Union of exp * exp

```
    env ▹ s1 ⟹ Set(t1, List_val l1)
    env ▹ s2 ⟹ Set(t2, List_val l2)
    env ▹ t1 = t2
    env ▹ l ⟹ l1 U l2
-------------------------------------------
    env ▹ Union(s1, s2) ⟹ Set(t1, List_val l)
```

Inter of exp * exp

$$\frac{\begin{array}{l}\texttt{env} \triangleright \texttt{s1} \Longrightarrow \texttt{Set(t1, List\_val l1)} \\ \texttt{env} \triangleright \texttt{s2} \Longrightarrow \texttt{Set(t2, List\_val l2)} \\ \texttt{env} \triangleright \texttt{t1} = \texttt{t2} \\ \texttt{env} \triangleright \texttt{l} \Longrightarrow \texttt{l1} \cap \texttt{l2}\end{array}}{\texttt{env} \triangleright \texttt{Inter(s1, s2)} \Longrightarrow \texttt{Set(t1, List\_val l)}}$$

Diff of exp * exp

$$\frac{\begin{array}{l}\texttt{env} \triangleright \texttt{s1} \Longrightarrow \texttt{Set(t1, List\_val l1)} \\ \texttt{env} \triangleright \texttt{s2} \Longrightarrow \texttt{Set(t2, List\_val l2)} \\ \texttt{env} \triangleright \texttt{t1} = \texttt{t2} \\ \texttt{env} \triangleright \texttt{l} \Longrightarrow \texttt{l1} - \texttt{l2}\end{array}}{\texttt{env} \triangleright \texttt{Diff(s1, s2)} \Longrightarrow \texttt{Set(t1, List\_val l)}}$$

Add of exp * exp

$$\frac{\begin{array}{l}\texttt{env} \triangleright \texttt{s} \Longrightarrow \texttt{Set(t, List\_val [v1, ..., vn])} \\ \texttt{env} \triangleright \texttt{elt} \Longrightarrow \texttt{v}\end{array}}{\texttt{env} \triangleright \texttt{Add(s, elt)} \Longrightarrow \texttt{Set(t, List\_val([v1, ..., vn] U v))}}$$

Remove of exp * exp

$$\frac{\texttt{env} \triangleright \texttt{s} \Longrightarrow \texttt{Set(t, List\_val [v1, ..., vn])} \quad \texttt{env} \triangleright \texttt{elt} \Longrightarrow \texttt{v}}{\texttt{env} \triangleright \texttt{Remove(s, elt)} \Longrightarrow \texttt{Set(t, List\_val ([v1, ..., vn] - v))}}$$

IsEmpty of exp

$$\frac{\texttt{env} \triangleright \texttt{s} \Longrightarrow \texttt{Set(t, List\_val [v1, ..., vn])}}{\texttt{env} \triangleright \texttt{IsEmpty(s)} \Longrightarrow \texttt{Bool(false)}}$$

$$\frac{\texttt{env} \triangleright \texttt{s} \Longrightarrow \texttt{Set(t, List\_val [])}}{\texttt{env} \triangleright \texttt{IsEmpty(s)} \Longrightarrow \texttt{Bool(true)}}$$

Contains of exp * exp

$$\frac{\texttt{env} \triangleright \texttt{s} \Longrightarrow \texttt{Set(t, List\_val [v1, ..., vn])} \quad \texttt{env} \triangleright \texttt{elt} \Longrightarrow \texttt{v env} \triangleright \texttt{v} \in \texttt{[v1, ..., vn]}}{\texttt{env} \triangleright \texttt{Contains(s, elt)} \Longrightarrow \texttt{Bool(true)}}$$

$$\frac{\texttt{env} \triangleright \texttt{s} \Longrightarrow \texttt{Set(t, List\_val [v1, ..., vn])} \quad \texttt{env} \triangleright \texttt{elt} \Longrightarrow \texttt{v env} \triangleright \texttt{v} \notin \texttt{[v1, ..., vn]}}{\texttt{env} \triangleright \texttt{Contains(s, elt)} \Longrightarrow \texttt{Bool(false)}}$$

Subset of exp * exp

$$\frac{env \triangleright s1 \Longrightarrow Set(t1, List\_val\ l1) \quad env \triangleright s2 \Longrightarrow Set(t2, List\_val\ l2) \quad env \triangleright t1 = t2 \quad env \triangleright l1 \not\subseteq l2}{env \triangleright Subset(s1, s2) \Longrightarrow Bool(false)}$$

$$\frac{env \triangleright s1 \Longrightarrow Set(t1, List\_val\ l1) \quad env \triangleright s2 \Longrightarrow Set(t2, List\_val\ l2) \quad env \triangleright t1 = t2 \quad env \triangleright l1 \subseteq l2}{env \triangleright Subset(s1, s2) \Longrightarrow Bool(true)}$$

MaxElt of exp

$$\frac{env \triangleright s \Longrightarrow Set(t, List\_val\ l) \quad env \triangleright t \Longrightarrow (\text{“int”} \vee \text{“string”})}{env \triangleright MaxElt(s) \Longrightarrow Max(l)}$$

MinElt of exp

$$\frac{env \triangleright s \Longrightarrow Set(t, List\_val\ l) \quad env \triangleright t \Longrightarrow (\text{“int”} \vee \text{“string”})}{env \triangleright MaxElt(s) \Longrightarrow Min(l)}$$

For_all of exp * exp

$$\frac{env \triangleright s \Longrightarrow Set(t, List\_val\ l) \quad env \triangleright p \Longrightarrow Fun(targ{:}t, tres{:}\text{"bool"}) \quad env \triangleright \forall x \in l \mid p(x) = Bool(true)}{env \triangleright For\_all(p, s) \Longrightarrow Bool(true)}$$

$$\frac{env \triangleright s \Longrightarrow Set(t, List\_val\ l) \quad env \triangleright p \Longrightarrow Fun(targ{:}t, tres{:}\text{"bool"}) \quad env \triangleright \exists x \in l \mid p(x) = Bool(false)}{env \triangleright For\_all(p, s) \Longrightarrow Bool(false)}$$

Exists of exp * exp

$$\frac{env \triangleright s \Longrightarrow Set(t, List\_val\ l) \quad env \triangleright p \Longrightarrow Fun(targ{:}t, tres{:}\text{"bool"}) \quad env \triangleright \exists x \in l \mid p(x) = Bool(true)}{env \triangleright Exists(p, s) \Longrightarrow Bool(true)}$$

$$\frac{env \triangleright s \Longrightarrow Set(t, List\_val\ l) \quad env \triangleright p \Longrightarrow Fun(targ{:}t, tres{:}\text{"bool"}) \quad env \triangleright \nexists x \in l \mid p(x) = Bool(true)}{env \triangleright Exists(p, s) \Longrightarrow Bool(false)}$$

Filter of exp * exp

```
env ▷ s ⟹ Set(t, List_val l)
env ▷ p ⟹ Fun(targ:t, tres:"bool")
env ▷ lr = l - (x ∈ l | p(x) = Bool(false))
---------------------------------------------
env ▷ Filter(p, s) ⟹ Set(t, List_val lr)
```

Map of exp * exp

```
env ▷ s ⟹ Set(t, List_val [v1, ..., vn])
env ▷ p ⟹ Fun(targ:t, tres:any_type)
---------------------------------------
env ▷ Map(p, s) ⟹ List_val [p(v1), ..., p(vn)]
```