System And Languages for Informatics
Cybersecurity LM-66

Samuele Padula

Gennaio 2021

1 Introduzione

È stato realizzato un interprete in OCaml per un ridotto linguaggio di programmazione funzionale. L'intero interprete con solamente typechecking dinamico e comprensivo dei test effettuati, è contenuto in un unico file sorgente "interprete_dinamico.ml". Nel file "interprete_statico.ml" è stato invece sviluppato l'interprete con typechecking statico rimuovendo tutti i controlli di tipo dinamici compiuti in precedenza dalla 'eval'. L'interprete visto a lezione è stato esteso con la possibilità di creare e manipolare gli insiemi. Gli insiemi possono contenere solamente Interi, Stringhe e Booleani. Per eseguire la batteria di test si può usare il comando ocaml (il tutto è stato simulato e sviluppato in ambiente Linux Ubuntu 18.04 ocaml 4.05):

```
$ocaml interprete_dinamico.ml
ocaml interprete_statico.ml
```

2 Regole operazionali (Set)

2.1 Introduzione del tipo di dato Set

$$\frac{env \rhd e \implies t : String \quad t \in \{"int", "bool", "string"\}}{env \rhd \mathbf{EmptySet(e)} \implies Set(t, \emptyset)}$$

$$\frac{env \rhd e_1 \implies t : String \quad t \in \{"int", "bool", "string"\}, env \rhd e_2 \implies v : t}{env \rhd \mathbf{Singleton(e_1, e_2)} \implies Set(t, v)}$$

$$\frac{env \rhd e_1 \implies t : String \quad t \in \{"int", "bool", "string"\}, env \rhd e_2 \implies (v_1 ... v_n) : t}{env \rhd \mathbf{Of(e_1, e_2)} \implies Set(t, \{v_1 ... v_n\})}$$

2.2 Operazioni sul tipo di dato Set

2.2.1 Operazioni di base

$$\frac{env \rhd e_1 \implies Set(t, s1), env \rhd e_2 \implies Set(t, s2) \quad t \in \{"int", "bool", "string"\}}{env \rhd \mathbf{Union(e_1, e_2)} \implies Set(t, \{s_1 \cup s_2\})}$$

$$\frac{env \rhd e_1 \implies Set(t, s1), env \rhd e_2 \implies Set(t, s2) \quad t \in \{"int", "bool", "string"\}}{env \rhd \mathbf{Intersection(e_1, e_2)} \implies Set(t, \{s_1 \cap s_2\})}$$

$$\frac{env \rhd e_1 \implies Set(t, s1), env \rhd e_2 \implies Set(t, s2) \quad t \in \{"int", "bool", "string"\}}{env \rhd \mathbf{Difference(e_1, e_2)} \implies Set(t, \{s_1 \setminus s_2\})}$$

2.2.2 Operazioni aggiuntive

$$\frac{env \rhd e_1 \implies Set(t, s), env \rhd e_2 \implies v : t \quad t \in \{"int", "bool", "string"\}}{env \rhd \mathbf{Insert(e_1, e_2)} \implies Set(t, s \cup \{v\})}$$

$$\frac{env \rhd e_1 \implies Set(t, s), env \rhd e_2 \implies v : t \quad t \in \{"int", "bool", "string"\}}{env \rhd \mathbf{Remove(e_1, e_2)} \implies Set(t, s \setminus \{v\})}$$

$$\frac{env \rhd e_1 \implies Set(t, s), env \rhd e_2 \implies v : t \quad t \in \{"int", "bool", "string"\}}{env \rhd \mathbf{Contains(e_1, e_2)} \implies Bool(v \in s)}$$

$$\frac{env \rhd e_1 \implies Set(t, s), env \rhd e_2 \implies v : t \quad t \in \{"int", "bool", "string"\}}{env \rhd \mathbf{Subset(e_1, e_2)} \implies Bool(\forall x \in s)}$$

$$\frac{env \rhd e \implies Set(t, s) \quad t \in \{"int", "bool", "string"\}}{env \rhd \mathbf{IsEmpty(e)} \implies Bool(s = \emptyset)}$$

$$\frac{env \rhd e \implies Set(t, s) \quad t \in \{"int", "bool", "string"\}}{env \rhd \mathbf{MinOf(e)} \implies min(s) : t}$$

$$\frac{env \rhd e \implies Set(t, s) \quad t \in \{"int", "bool", "string"\}}{env \rhd \mathbf{MaxOf(e)} \implies max(s) : t}$$

$$\frac{env \rhd e_1 \implies Set(t, s), env \rhd e_2 \implies f : t \to Boolean \quad t \in \{"int", "bool", "string"\}}{env \rhd \mathbf{ForAll(e_1, e_2)} \implies Bool(\forall x \in s, Apply(e_2, x) \implies Bool(true))}$$

$$\frac{env \rhd e_1 \implies Set(t, s), env \rhd e_2 \implies f : t \to Boolean \quad t \in \{"int", "bool", "string"\}}{env \rhd \mathbf{Exists(e_1, e_2)} \implies Bool(\exists x \in s, Apply(e_2, x) \implies Bool(true))}$$

$$\frac{env \rhd e_1 \implies Set(t, s), env \rhd e_2 \implies f : t \to Boolean \quad t \in \{"int", "bool", "string"\}}{env \rhd \mathbf{Filter(e_1, e_2)} \implies Set(t, \{x \in s, Apply(e_2, x) \implies Bool(true)\}}$$

$$\frac{env \rhd e_1 \implies Set(t_1, s), env \rhd e_2 \implies f : t_1 \to t_2 \quad t_1, t_2 \in \{"int", "bool", "string"\}}{env \rhd \mathbf{Map(e_1, e_2)} \implies Set(t_2, \{Apply(e_2, x \in s)\}}$$