Homework Assignment 1

Federico Bernacca

1 Introduzione

Per svolgere l'homework ho effettuato le seguenti scelte progettuali.
L'interprete usa l'ambiente come stack per i legami tra variabili e valori più un'altra struttura che mantiene al suo interno i permessi concessi durante l'esecuzione, e che una funzione deve avere per essere dichiarata.
L'interprete dunque avrà un security manager da invocare che fa parte del runtime.

2 Estensioni

Di seguito le estensioni fatte all'interprete per permettere il controllo degli accessi.

2.1 Permessi

Ho definito un nuovo costrutto sintattico per definire i permessi concessi durante l'esecuzione e richiesti da una funzione.

```
type permission =
    | None
    | Read
    | Write
    | Permission of permission * permission (* list of permission *)
```

2.2 Espressioni

Ho deciso di modificare il costrutto sintattico delle espressioni in modo che, durante la dichiarazione di una funzione, vengano passati anche i permessi da lei necessari per operare.
Inoltre, ho incluso la possibilità di abilitare e disabilitare diversi permessi per una particolare esecuzione.

```
type exp =
    ...
    (* formal parameter with function body and permissions *)
    | Fun of string * exp * permission
    (* name to assign to fun exp, new permission(s), exp in which evaluate *)
    | EnableIn of string * exp * permission * exp
    (* name to assign to fun exp, permission(s) to remove, exp in which evaluate *)
    | DisableIn of string * exp * permission * exp
    ...
```

Ho definito un nuovo tipo sintattico per il costrutto primitivo che effettua il controllo dei permessi delle funzioni piuttosto che inserirlo direttamente in *exp* perché l'idea è che in questo modo sia invisibile al programmatore.

```
type iexp =
  | Check of exp;; (* exp must be Fun *)
```

2.3 Interprete

Ho esteso l'interprete, quindi la funzione *eval* per permettere il passaggio dei permessi abilitati durante l'esecuzione, modificato il caso in cui *exp* sia di tipo *Fun*, e aggiunto i due costrutti per abilitare o disabilitare particolari permessi.

```
let rec eval (exp: exp) (env: 'v env) (gp: permission): value =
    match exp with
    ...
    | Fun(_, _, _) -> ieval (Check(exp)) gp env
    | EnableIn(ide, f, p, body) -> ...
    | DisableIn(ide, f, p, body) -> ...
    ...
```

Dove la funzione

```
let ieval (iexp: iexp) (p: permission) (env: 'v env): value
```

rappresenta il security manager che controlla se i permessi richiesti dalla funzione sono concessi; in caso di esito positivo restituisce la chiusura, altrimenti solleva un'eccezione.