Relazione Homework Assignment 2

Fausto Francesco Frasca 559482

May 2021

1 Scelte progettuali

Nell'estensione del linguaggio didattico sono stati implementati dei costrutti che permettono al programmatore di definire le opportune politiche da adottare all'interno di frame in base alla storia di esecuzione del programma. Ogni politica è definita come un automa a stati finiti e si basa su un alfabeto locale. Tale approccio permette una maggiore flessibilità al programmatore, poiché dovrà definire un insieme sigma che può essere un sottoinsieme di tutti i possibili simboli rappresentanti le azioni di sicurezza più rilevanti che possono essere eseguite dal programma. Nel caso specifico, l'insieme di tutte le operazioni di sicurezza rilevanti è costituito dal *Read, Write* e *Connect*. Un automa può avere un insieme sigma contenente solo Read e Write e tutte le possibili transizioni potranno contenere solo quei determinati simboli.

Tutte quelle operazioni che non fanno parte dell'insieme sigma verranno scartate, facendo permanere l'automa nello stato in cui si trova. Continuando con l'esempio riportanto in precedenza, se il programma esegue una Connect e l'automa si trova nello stato iniziale, quest'ultimo continuerà a permanere in tale stato.

Inoltre il linguaggio è stato esteso per supportare le funzioni ricorsive.

2 Implementazione

2.1 Tipi

Sono stati introdotti nuovi tipi, oltre a quelli già presenti nel linguaggio didattico. I più rilevanti sono:

- *dfa*: Ogni politica definita dal programmatore è di questo tipo.

  Ogni politica è una quintupla contenente:

  - Lista degli stati
  - Insieme dei simboli
  - Stato iniziale
  - Insieme di tutte le possibili transizioni

  - Insieme degli stati di accettazione

- *policy_stack*: Definisce il tipo dello stack contenente tutte le politiche in esecuzione a run-time. Ogni frame dello stack è di tipo *policy_frame*, ed è una coppia (stato corrente, dfa).

- *history_list*: Definisce il tipo della lista contenente tutta la storia d'esecuzione del programma. Ogni elemento nella lista è di tipo *symbol*, stesso tipo dei simboli contenuti nell'insieme sigma presente in ogni politica.

2.2 Funzioni ausiliarie

Sono state aggiunte delle funzioni per il supporto alle politiche e alla loro esecuzione. Le principali sono:

- *well_formed_dfa*: Controlla che la politica passata in input sia ben formata. Se tutti i simboli presenti nella lista delle transizioni appartengono all'insieme sigma e gli stati di partenza e di accettazioni appartengo all'insieme degli stati allora ritorna true, false altrimenti.

- *check_history*: Prende in input una determinata politica le la history del programma e ritorna la coppia (end_state, accepted). Il primo parametro indica lo stato in cui si trova l'automa dopo aver eseguito tutte le operazioni presenti nella history. Il secondo è un booleano che indica se end_state è uno stato di accettazione o meno.

- *check_policy_stack*: Presi in input l'attuale policy stack e il simbolo da eseguire, aggiorna lo stato corrente di tutte le policy in accordo alla loro funzione di transizione. Lancia un'eccezione se una delle policy viene violata, altrimenti ritorna un nuovo policy stack col nuovo stato corrente di tutte le policy.

2.3 Funzione di valutazione

È stata opportunamente modificata la funzione di valutazione *eval*. La funzione adesso prende in input, oltre all'espressione da valutare e l'ambiente, uno stack per le policy, di tipo policy_stack, e una lista per tenere traccia della storia del programma, di tipo history_list. Ogni nuova operazione rilevante per la history viene posta in fondo alla lista. Il valore di ritorno della funzione è di tipo *ret_val*. Tale tipo è una tripla contenente il valore della valutazione dell'espressione, lo stack delle politiche e la lista history.

Per quanto riguarda i nuovi costrutti gestisti dalla funzione sono:

- *RecFun*: Costrutto per le funzioni ricorsive.

- *Frame*: Costrutto che definisce un frame a cui applicare una determinata politica. Prima di eseguire il corpo del frame viene richiamata la funzione well_formed_dfa sulla policy passata in input. Se la policy è ben formata,

viene aggiornato lo stato corrente dell'automa in base alla history del programma richiamando la funzione check_history. Se la funzione ritorna esito positivo vuol dire che la policy non è stata violata e si prosegue con la valutazione del corpo del frame.

- *Read, Write, Connect*: Queste operazioni rappresentano le operazioni rilevanti nell'esecuzione di un programma. Tali operazioni perciò sono le stesse che possono essere contenute nell'insieme sigma di una data politica e che saranno contenute nella history del programma.

3 Esempi

Sono state definite due politiche. La politica *no_write* è definita sull'insieme sigma(Read, Write). Se viene effetuata una write, l'automa finirà in uno stato di non accettazione. La seconda politica è quella vista a lezione "No read after write", anch'essa definita sul medesimo insieme sigma.

Per quanto riguardo il primo esempio, questo rappresenta l'esecuzione in sequenza di

$$Connect, no\_read\_after\_write[Read, no\_write[Read], Write], Write$$

Il programma termina senza alcuna eccezione poiché nessuna politica viene violata. Il secondo esempio invece riporta l'esecuzione in sequenza

$$Read, no\_read\_after\_write[Read], Write, no\_read\_after\_write[Read]$$

In questo caso il programma termina con un'eccezine poiché l'unica politica adottata viene violata.