UNR Universidad Nacional de Rosario

Facultad de Cs. Exactas, Ingeniería y Agrimensura

Licenciatura en Ciencias de la Computación

Ingeniería de Software

Trabajo Práctico: Verificación de Software

Mercedes Castro
Legajo: C-6396/7

1. Requerimientos

Una base de datos almacena información sobre películas. La misma debe ser capaz de guardar quien dirigió un film. Todo film en la base de datos debe tener asociado un director.

Mediante una consulta a la base de datos se debe poder encontrar el director de una determinada película. Así como también todas las películas dirigidas por una persona en particular.

Además se cuenta con operaciones para efectuar altas y bajas de peliculas.

2. Especificación

Designaciones:

> t es un título de película $\approx t \in TITLE$
>
> n es un nombre $\approx n \in NAME$
>
> m es el título de una película cuyo director hay registrar $\approx m \in movies$
>
> El director de una película $m \approx director\ m$

Tipos básicos:

$$[TITLE, NAME]$$

Espacio de estados de la especificación:

Database

$$movies : \mathbb{P}\ TITLE$$
$$director : TITLE \pfun NAME$$

Estado inicial de la base de datos:

DatabaseInit

$$Database$$

$$movies = \emptyset$$
$$director = \emptyset$$

Invariantes de estado:

DatabaseInv

$$Database$$

$$movies = \text{dom}\ director$$

Primera Operación: Alta de una película con su respectivo director en la base de datos.

Caso Exitoso:

AddFilmOk

$$\Delta Database$$
$$t? : TITLE$$
$$dir? : NAME$$

$$t? \notin movies$$
$$movies' = movies \cup \{t?\}$$
$$director' = director \cup \{t? \mapsto dir?\}$$

Caso de error:

TitleAlreadyExists

$$\Xi Database$$
$$t? : TITLE$$

$$t? \in movies$$

$$AddFilm == AddFilmOk \lor TitleAlreadyExists$$

Segunda Operación: Baja de una película en la base de datos.

Caso exitoso:

RemoveFilmOk

$$\Delta Database$$
$$t? : TITLE$$

$$t? \in movies$$
$$movies' = movies \setminus \{t?\}$$
$$director' = director \setminus \{t? \mapsto director\ t?\}$$

Caso de error:

TitleDoesNotExist

$$\Xi Database$$
$$t? : TITLE$$
$$t? \notin movies$$

$$RemoveFilm == RemoveFilmOk \lor TitleDoesNotExist$$

Tercera operación: Mostrar el director de una película determinada.

Caso Exitoso:

FindDirectorOk

$$\Xi Database$$
$$t? : TITLE$$
$$d! : NAME$$
$$t? \in movies$$
$$d! = director\ t?$$

Caso de error: TitleDoesNotExist

$$FindDirector == FindDirectorOk \lor TitleDoesNotExist$$

Cuarta Operación: Lista los nombres de las películas con un determinado director.

SameDirector

$$\Xi Database$$
$$d? : NAME$$
$$films! : \mathbb{P}\ TITLE$$
$$films! = \mathrm{dom}(director \rhd \{d?\})$$

3. Simulaciones

La primera simulación es la siguiente:

```
databaseInit(S0) & addFilm(S0,title1,director1,S1) &
addFilm(S1,title2,director2,S2) & addFilm(S2,title3,director1,S3) &
removeFilm(S3,title2,S4) & sameDirector(S4,director1,Movies,S5).
```

cuya primera respuesta es la siguiente:

```
S0 = {[movies,{}],[directors,{}]},
S1 = {[movies,{title1}],[directors,{[title1,director1]}]},
S2 = {[movies,{title1,title2}],[directors,{[title1,director1],
      [title2,director2]}]},
S3 = {[movies,{title1,title2,title3}],[directors,{[title1,director1],
      [title2,director2],[title3,director1]}]},
S4 = {[movies,{title1,title3}],[directors,{[title1,director1],
      [title3,director1]}]},
Movies = {title1,title3},
S5 = {[movies,{title1,title3}],[directors,{[title1,director1],
      [title3,director1]}]}
```

La segunda simulación es la siguiente:

```
S0 = {[movies,{title1,title2}],[directors,{[title1,director1],
[title2,director2]}]} & findDirector(S0,title2,Output,S1) &
removeFilm(S1,title2,S2) &  findDirector(S2,title2,Output2,S3).
```

cuya primera respuesta es la siguiente:

```
S0 = {[movies,{title1,title2}],[directors,{[title1,director1],
[title2,director2]}]},
Output = director2,
S1 = {[movies,{title1,title2}],[directors,{[title1,director1],
[title2,director2]}]},
S2 = {[movies,{title1}],[directors,{[title1,director1]}]},
S3 = {[movies,{title1}],[directors,{[title1,director1]}]}
```

4. Demostraciones con $\{log\}$

Primera demostración con $\{log\}$. Demuestro que *AddFilm* preserva el invariante *DatabaseInv*, o sea el siguiente teorema:

theorem AddFilmPI
$DatabaseInv \wedge AddFilm \Rightarrow DatabaseInv'$

el cual en $\{log\}$ se escribe de la siguiente forma:

```
Database = {[movies,M],[directors,D]} &
Database_ = {[movies,M_],[directors,D_]} &
dom(D,M) &
addFilm(Database,T,Dir,Database_) &
ndom(D_,M_).
```

Segunda demostración con $\{log\}$. Demuestro que *RemoveFilm* preserva el invariante *DatabaseInv*, o sea el siguiente teorema:

theorem RemoveFilmPI
$DatabaseInv \wedge RemoveFilm \Rightarrow DatabaseInv'$

el cual en $\{log\}$ se escribe de la siguiente forma:

```
Database = {[movies,M],[directors,D]} &
Database_ = {[movies,M_],[directors,D_]} &
dom(D,M) &
removeFilm(Database,T,Database_) &
ndom(D_,M_).
```

5. Demostración con Z/EVES

theorem AddFilmPI
$DatabaseInv \land AddFilm \Rightarrow DatabaseInv'$

proof[*AddFilmPI*]
invoke AddFilm;
split AddFilmOk;
cases;
prove by reduce;
next;
prove by reduce;
next;
■

6. Casos de prueba

El script que usé para generar casos de prueba con Fastest es el siguiente:

```
loadspec fastest.tex
selop AddFilm
genalltt
addtactic AddFilm_DNF_1 SP \cup director \cup \{t? \mapsto dir?\}
genalltt
genalltca
```

Es decir que generé casos de prueba para la operación *AddFilm*. Aplico primero DNF sobre la operación lo cual me genera las clases de prueba para el caso exitoso y el caso erróneo. Dado que me interesa generar casos de prueba solo para el primero, aplico SP sobre la expresión $birthday \cup \{name? \mapsto date?\}$ pero solo para particionar la clase de prueba *AddBirthday_DNF_*1.

De esta forma Fastest generó el siguiente árbol de clases de prueba satisfacibles y sus respectivos casos:

```
AddFilm_VIS
  !____AddFilm_DNF_1
  |    !____AddFilm_SP_2
  |    |    !____AddFilm_SP_2_TCASE
  |    |
  |    !____AddFilm_SP_4
  |    |    !____AddFilm_SP_4_TCASE
  |    |
  |    !____AddFilm_SP_6
  |    |    !____AddFilm_SP_6_TCASE
  |    |
  |    !____AddFilm_SP_7
  |    |    !____AddFilm_SP_7_TCASE
  |    |
  |    !____AddFilm_SP_8
  |
  !____AddFilm_DNF_2
       !____AddFilm_DNF_2_TCASE
```

Los casos de prueba son los siguientes:

AddFilm_SP_2_TCASE

AddFilm_SP_2

$director = \emptyset$
$dir? = name3$
$movies = \{title1\}$
$t? = title2$

AddFilm_SP_4_TCASE

AddFilm_SP_4

$director = \{(title2 \mapsto name3)\}$
$dir? = name5$
$movies = \{title4\}$
$t? = title1$

AddFilm_SP_6_TCASE

AddFilm_SP_6

$director = \{(title1 \mapsto name2), (title3 \mapsto name4)\}$
$dir? = name2$
$movies = \{title5\}$
$t? = title1$

AddFilm_SP_7_TCASE

AddFilm_SP_7

$director = \{(title1 \mapsto name2)\}$
$dir? = name2$
$movies = \{title3\}$
$t? = title1$

AddFilm_DNF_2_TCASE

AddFilm_DNF_2

$director = \emptyset$
$dir? = name2$
$movies = \{title1\}$
$t? = title1$