

Falluto: Un model checker para la verificación de sistemas tolerantes a fallas

Edgardo E. Hames
Director: Dr. Pedro R. D'Argenio

FaMAF - UNC

14 de diciembre de 2009

1985 – 1987, EEUU – Canadá



200 veces más radiación

6 muertos

1996, Guayana Francesa



37 segundos



7.500 millones



1997, Marte



súbitos reinicios del sistema

pérdida de datos



2007, Salt Lake City

1 día



566 vuelos retrasados



2009, New Hampshire



resumen de cuenta

US\$ 23.148.855.308.184.500

mayor que el PBI del mundo


problemas en el software

no hay que preocuparse

1 cada 10 años

1947

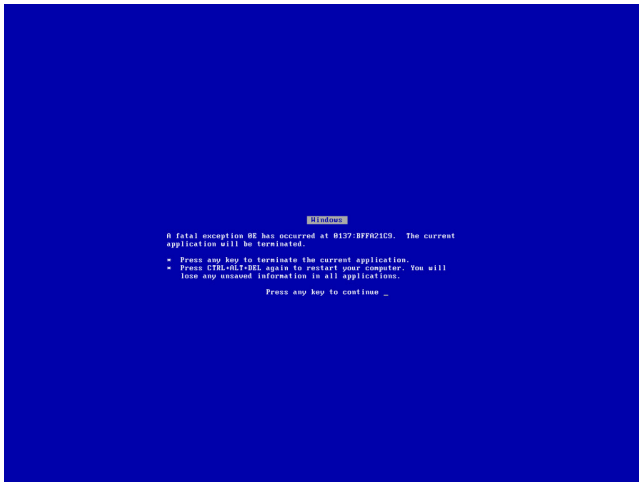
primer *bug*

Andam started
 " stopped - andam ✓
 1300 (032) HP-MC ~~1.982677000~~
 (033) PRO 2 2.130476415
 convd 2.130676415
 Relays 6-2 in 033 failed special speed test
 in relay - 10.000 test.
 Relays changed
 Started Cosine Tape (Sine check)
 Started Multi-Adder Test.

 Relay #70 Panel
 (moth) in relay.
 First actual case of bug being found.
 Andam started.
 closed down.

fallas

internas


división por cero





externas



Andam started
 " stopped - andam ✓
 1300 (032) HP-MC ~~1.982677000~~
 (033) PRO 2 2.130476415
 convd 2.130676415
 Relays 6-2 in 033 failed special speed test
 in relay - 10.000 test.
 Relays changed
 Started Cosine Tape (Sine check)
 Started Multi-Adder Test.

 Relay #70 Panel
 (moth) in relay.
 First actual case of bug being found.
 Andam started.
 closed down.

quiero demostrar la
corrección de mi programa

¿qué es programa?

conjunto finito de acciones

$\text{nombre} :: \langle \text{guarda} \rangle \rightarrow \langle \text{comando} \rangle$

¿qué es corrección?

programa satisface especificación

$$p \models \psi$$

¿qué es especificación?

definición de un problema

dos propiedades

safety

“no ocurre algo malo”

liveness

“algo bueno ocurre”

un ejemplo cotidiano

cruzar la calle

$G \neg(\text{el mono es atropellado})$



F (el mono cruza)



quiero demostrar la
corrección de mi programa

ensayos

“puede revelar la presencia de errores,
pero nunca demostrar su ausencia”

– E. Dijkstra, 1976

model checking

método formal

verificación automática

$$p \models \psi$$

el programa es correcto

$$p \not\equiv \psi$$

contraejemplo

corregimos el programa

con model checking

se come, se educa y se cura

¿y las fallas?

quiero demostrar la
corrección de mi programa
en entornos con fallas

¿qué es falla?

una acción no deseada

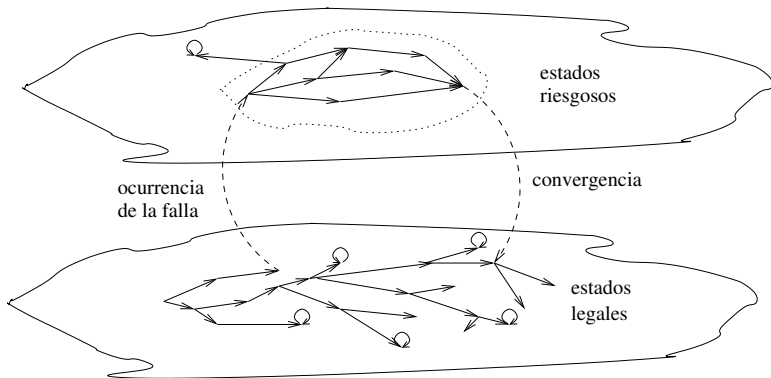
modelo de fallas

transforma nuestro programa

tolerancia a fallas

programa transformado
satisface su especificación

$$F(p) \models \psi$$



quiero demostrar la
corrección de mi programa
en entornos con fallas

Falluto

model checker para verificar sistemas tolerantes a fallas

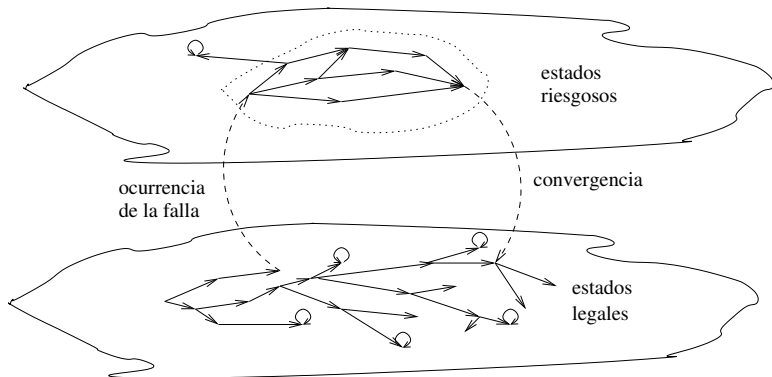
lenguaje declarativo

extiende el lenguaje de NuSMV
(otro model checker)

agrega declaración de fallas

FAULT "nombre"

```
pre( cuando puede ocurrir );  
effect( lista de efectos );  
restores( cuando se recupera );
```



crash

FAULT Crash

```
pre(TRUE);  
effect();  
restores(FALSE);
```

agrega inhabilitación de
acciones

<acción>

```
<acción> disabled_by { lista de fallas };
```


`next(x) := x + 1;`

```
next(x) := x + 1 disabled_by { Crash };
```

¿cómo funciona?

compila a código de NuSMV

crea un proceso por falla

variable *active*

$$active \Rightarrow pre$$

cambia valores de variables
(effect)

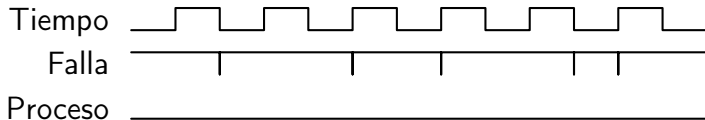
$$\text{restores} \Rightarrow \neg \text{active}$$

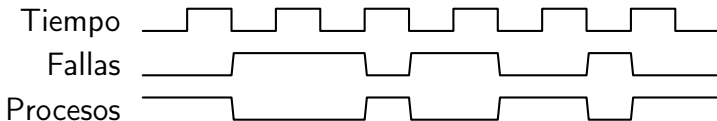
transformamos las
asignaciones originales

```
next(x) := x + 1 disabled_by { crash };
```

```
next(x) :=  
  case  
    !crash.active : x + 1;  
    1              : x;  
  esac;
```

problemas de equidad





variable de equidad $vfair_f$

fuerza la ejecución del proceso

```
next(y) :=  
  case  
    y < 10 : y + 1 disabled_by {f};  
    1      : y;  
  esac;
```

```
next(y) :=  
  case  
    y < 10 : y + 1 disabled_by {f};  
    1      : y;  
  esac;
```

```
next(vfair_f) :=  
  case  
    y < 10 & !f.active & !vfair_f : 1;  
    1                             : 0;  
  esac;
```

equidad fuerte entre procesos y fallas

condicionamos la especificación

$$\mathbf{GF}(y < 10 \wedge \neg f.active) \Rightarrow \mathbf{GF}(vfair_f)$$

$$F(p) \models (\textit{equidad fuerte} \Rightarrow \psi)$$

modelos de fallas

fail-stop

omisión de envío

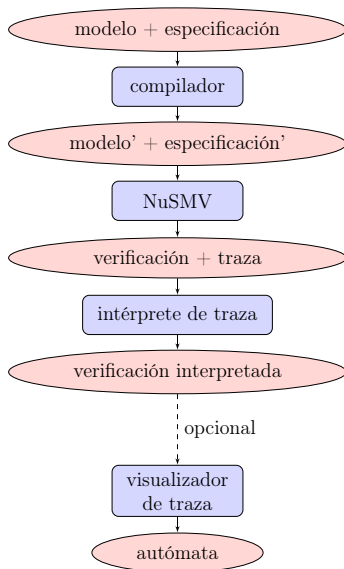
omisión de recepción

bizantina

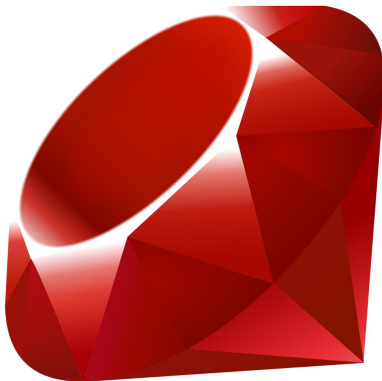
enlace

reinicio

arquitectura



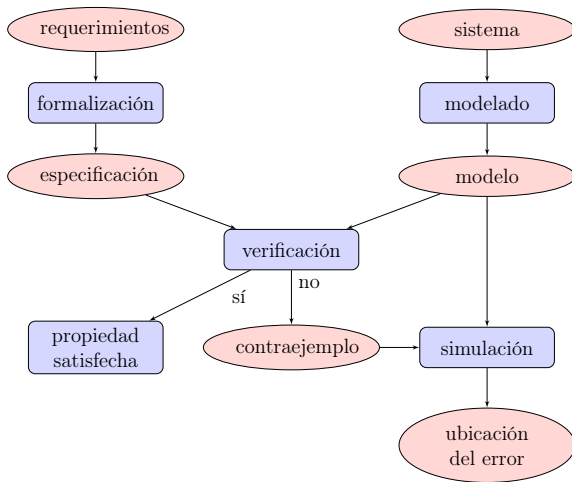
<chivo>

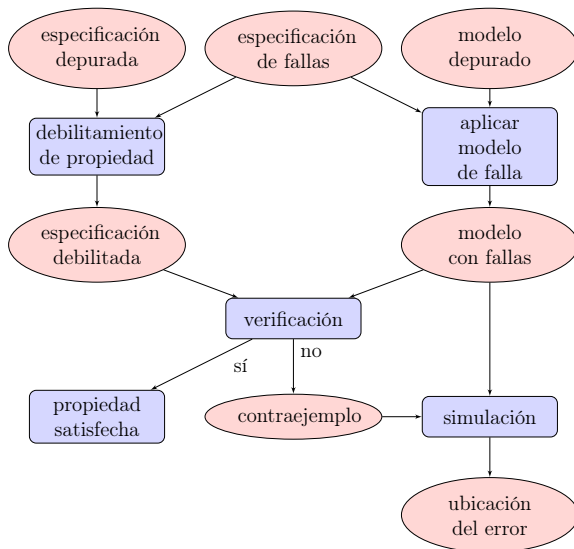




</chivo>

¿cómo se usa?





commit atómico

elección de líder

lenguaje declarativo

simplifica el modelado

permite encontrar errores

trabajos futuros

meta-propiedades predefinidas

nuevos tipos de fallas

nueva sintaxis

¿preguntas?

¡gracias por venir!