

Falluto2.0.

Un model checker para sistemas tolerantes a fallas.

R. Monti P. D'Argenio N. Aguirre

Facultad de Matemática, Astronomía y Física, Universidad Nacional de
Córdoba

Córdoba, Argentina. 2013

Tabla de contenidos

- 1 Introducción
- 2 Sistemas de transición de estados
- 3 Lenguaje de Falluto2.0
- 4 Compilación de Falluto2.0
- 5 Fairness en Falluto2.0
- 6 Interpretación de trazas
- 7 Resultados
- 8 Ejemplo de uso

Tabla de contenidos

- 1 Introducción
- 2 Sistemas de transición de estados
- 3 Lenguaje de Falluto2.0
- 4 Compilación de Falluto2.0
- 5 Fairness en Falluto2.0
- 6 Interpretación de trazas
- 7 Resultados
- 8 Ejemplo de uso

Sistemas tolerantes a fallas

- Sistemas críticos.
- Fallas \neq Errores.
- Sistemas tolerantes a fallas.
- Model checking.

■ Sistemas críticos.

Ejemplos: aviónica, equipos médicos, etc...

■ Fallas \neq Errores.

Falla = cambio de estado interno.

Error = desviación del comportamiento esperado.

■ Sistemas tolerantes a fallas.

No podemos evitar las fallas \rightarrow las superamos.

Model Checking.

- Método formal para la verificación de propiedades sobre sistemas.
- Exhaustivo sobre el espacio de estados del modelo finito.
- NuSMV es un model checker basado en BDD.

Motivación

```
int y1 = 0;
int y2 = 0;
short in_critical = 0;
```

```
active proctype process_1() {
  do
    :: true ->
      y1 = y2+1;
      ((y2==0) || (y1<=y2));
      in_critical++;
      in_critical--;
      y1 = 0;
  od
}
```

```
active proctype process_2() {
  do
    :: true ->
      y2 = y1+1;
      ((y1==0) || (y2<y1));
      in_critical++;
      in_critical--;
      y2 = 0;
  od
}
```

Objetivos

- Continuar con el trabajo hecho en Falluto(Hames) y Offbeat(Bordenabe).
- Definir un lenguaje práctico para el modelado y verificación de sistemas tolerantes a fallas, que oculte las funcionalidad de las fallas y provea facilidades para la especificación de propiedades.
- Elaborar un front-end para **NuSMV** que use este lenguaje.

Tabla de contenidos

- 1 Introducción
- 2 Sistemas de transición de estados
- 3 Lenguaje de Falluto2.0
- 4 Compilación de Falluto2.0
- 5 Fairness en Falluto2.0
- 6 Interpretación de trazas
- 7 Resultados
- 8 Ejemplo de uso

Estructuras de Kripke

Una estructura de Kripke sobre AP se define como una 4-upla $M = (S, I, R, L)$ donde

- S es un conjunto finito de estados
- $I \subseteq S$ estados iniciales
- $R \subseteq S \times S$ relación de transición left-total.
- $L: S \rightarrow P(AP)$ función de etiquetado o interpretación.

En model checking usualmente sucede que:

- S es un conjunto de valuaciones sobre las variables del sistema.
- AP es un conjunto de expresiones booleanas sobre las variables.
- $L(v) = \{a \in AP \mid v(a) \text{ es verdadero}\}$ con $v \in S$
- R explica la relación entre la valuación actual y la próxima

Ejemplo de estructura de Kripke

LTS

Un LTS es una tres-upla $M = (S, S_0, L, R)$ donde:

- S es un conjunto de estados
- $S_0 \subseteq S$ es un conjunto de estados iniciales
- L es un conjunto de etiquetas (nombres de transiciones)
- $R \subseteq S \times L \times S$ es una relación ternaria de transiciones etiquetadas

Si $s_1, s_2 \in S$, $l \in L$, y $(s_1, l, s_2) \in R$, entonces existe una transición con nombre l desde el estado s_1 al estado s_2 .

Ejemplo de LTS

Tabla de contenidos

- 1 Introducción
- 2 Sistemas de transición de estados
- 3 Lenguaje de Falluto2.0**
- 4 Compilación de Falluto2.0
- 5 Fairness en Falluto2.0
- 6 Interpretación de trazas
- 7 Resultados
- 8 Ejemplo de uso

El lenguaje de NuSMV

Modelado

```

MODULE main()
  VAR
    v1:boolean;
    v2:{1,2,a,b};

  INIT
    v1 & v2 = a

  TRANS
    ( v1 -> next(v2) in {a,b}
    & !v1 -> next(v2) in {1,2}
    )
    | next(v1) = !v1

```

Propiedades

```

LTLSPEC G v1 = FALSE

CTLSPEC AG TRUE

```

Fairness

```

FAIRNESS v2 in {1,a}

COMPASSION (v2 = 1 , !v1)

```


El lenguaje de Falluto2.0

Modelado

```

PROCTYPE proc( ctxv1, ctxv2
                ; sync1, sync2)

  VAR
    var1 : bool

  INIT
    {formula}

  TRANS
    [name]: enable => changes
    [sync1] ...
    []: ...
ENDPROCTYPE

...

```

Instanciación y verificación

```

INSTANCE inst1 =
  proc(inst2.v,TRUE,s1,s1)

INSTANCE inst2 = proc2(s1)

LTLSPEC ...
CTLSPEC ...
NORMAL_BEHAIVIOUR ...

FAIRNESS ...
COMPASSION( ... )

```

Descripción de transiciones

`[nombre]: cond_habilitación => post_condición`

La condición de habilitación es una fórmula booleana sobre el estado actual del sistema.

Las post condiciones son listas de next- valores:

$$v1' = f1, v2' = f2, v3' = f3, \dots$$

Inyección de fallas

La inyección de fallas se realiza de modo declarativo, en la sección introducida por la palabra reservada **FAULT** de cada proctype:

```
nombre : cond_habilitación => post_condicin is TYPE
```

donde TYPE puede ser:

- **STOP**[(trans1, trans2, ...)]
- **BYZ**([var1, var2, ...])
- **TRANSIENT**

El lenguaje de Falluto2.0

Modelado

```
PROCTYPE machine(; transfer)
VAR
  on : bool
  file : 0..9
INIT
  !on & file = 0
TRANS
  [send]: on
    =>
      file' = (file+1)%10
  [OnOff]: => on' = !on
ENDPROCTYPE
```

Inyección de falla

```
PROCTYPE machine(; transfer)
VAR
  ...

FAULT
  f1: is STOP(OnOff)
  f2: file = 9 => is
      BYZ(file)
  f3: => file' = 0 is
      TRANSIENT

INIT ...
TRANS ...
ENDPROCTYPE
```

Especificación de propiedades

Propiedades como en NuSMV:

LTLSPEC q

CTLSPEC q

Propiedades sobre escenarios comunes

“Si solo hago buenas transiciones entonces q es verdadera”:

`NORMAL_BEHAVIOUR` $\rightarrow q$

“Si eventualmente las fallas dejan de ocurrir entonces q es verdadera”:

`FINITELY_MANY_FAULTS` $\rightarrow q$

Tabla de contenidos

- 1 Introducción
- 2 Sistemas de transición de estados
- 3 Lenguaje de Falluto2.0
- 4 Compilación de Falluto2.0**
- 5 Fairness en Falluto2.0
- 6 Interpretación de trazas
- 7 Resultados
- 8 Ejemplo de uso

Compilación de transiciones

Transiciones comunes:

[trans]: $pre \Rightarrow pos1, pos2, \dots$

Se compila como:

$$act\#var' = trans \wedge !stop \wedge pre \wedge pos1 \wedge pos2 \wedge \dots \wedge unchanged$$

Donde:

$$unchanged = \bigwedge_{v \in V} (v' = v) \text{ con } V = Vars - POS - act\#var$$

Compilación de transiciones

Transiciones sincronizadas:

Por cada sincronización tendremos:

$$act\#var' = syncname \wedge !STOP \wedge pre1 \wedge pre2 \wedge \dots \\ \wedge POS1 \wedge POS2 \wedge \dots \wedge unchanged$$

Donde

$$unchanged = \bigwedge_{v \in V} (v' = v) \text{ con } V = Vars - \bigcup (POS_j) - act\#var$$

STOP involucra las fallas que afectan cada una de las transiciones sincronizadas.

Compilación de transiciones

Transiciones de falla:

fault: $pre \Rightarrow pos1, pos2, \dots$ is Type

Compila a:

$$act\#var' = fault \wedge !fault_active \wedge pre \wedge fault_active' \\ \wedge pos1 \wedge pos2 \wedge \dots \wedge unchanged$$

$!fault_active$ y $fault_active'$ solo en caso de fallas permanentes.

Compilación de transiciones

Transición de deadlock:

$$act\#var' = dk\#trans \wedge neg_pre \wedge unchanged$$

$$neg_pre = \bigwedge_{i \in instances} \bigwedge_{t \in transN_i} (! pre_{i,t} \mid Stop_{i,t}) \\ \bigwedge_{s \in sincro} (! pre_s \mid Stop_s)$$

Donde:

$$unchanged = \bigwedge_{v \in V} (v' = v) \text{ con } v \in Var - act\#var$$

Compilación de transiciones

Verificación de deadlock

Usamos la propiedad `CHECK_DEADLOCK`.

Es compilada como:

$$CTLSPEC \ AG \ act\#var \ != \ dk\#trans$$

Notar relación con la transición de deadlock.

Compilación de transiciones

Compilación de la relación de transición final:

Consiste de la disyunción *exclusiva* de cada una de las transiciones del sistema:

$$\bigvee_{t \in T} t$$

Con $T = \text{transiciones_comunes} \cup \text{transiciones_sincronizadas} \cup \text{transiciones_de_falla} \cup \text{transicion_de_deadlock}$

Tabla de contenidos

- 1 Introducción
- 2 Sistemas de transición de estados
- 3 Lenguaje de Falluto2.0
- 4 Compilación de Falluto2.0
- 5 Fairness en Falluto2.0**
- 6 Interpretación de trazas
- 7 Resultados
- 8 Ejemplo de uso

Weak fairness entre procesos

$$G \text{ habilitada} \rightarrow G \text{ F atendida}$$

INST_WEAK_FAIR_DISABLE:

deshabilita la condición de fairness entre los procesos.

Compilación:

$$FAIRNESS (\bigvee_{t \in T_{N_i}} act \# var = t) \mid instanceDK$$

Donde $instanceDK = \bigwedge_{t \in T_i} ! pre_t$

Fairness con respecto a fallas

$$G \ F \ \textit{transicion_buena}$$

FAULT_FAIR_DISABLE:

deshabilita la condición de que a menudo se haga una transición normal.

Compilación:

$$FAIRNESS \ (\bigvee_{t \in T_N} act \# var = t) \mid act \# var = dk \# trans$$

Tabla de contenidos

- 1 Introducción
- 2 Sistemas de transición de estados
- 3 Lenguaje de Falluto2.0
- 4 Compilación de Falluto2.0
- 5 Fairness en Falluto2.0
- 6 Interpretación de trazas**
- 7 Resultados
- 8 Ejemplo de uso

Intérprete de trazas

```
-> State: 1.1 <-
action# = dk#action
lvar#inst#var1 = FALSE
lvar#inst#var2 = 0
ipc#inst = 0
-> State: 1.2 <-
action# = trans#inst#trans1
lvar#inst#var2 = 1
-> State: 1.3 <-
lvar#inst#var2 = 2
-> State: 1.4 <-
action# = trans#inst#trans2
lvar#inst#var1 = TRUE
ipc#inst = 1
```

```
---> State: 0 <---
inst var1 = FALSE
inst var2 = 0
```

```
@ [action] inst / trans1
---> State: 1 <---
inst var2 = 1
```

```
@ [action] inst / trans1
---> State: 2 <---
inst var2 = 2
```

```
@ [action] inst / trans2
---> State: 3 <---
inst var1 = TRUE
```

Tabla de contenidos

- 1 Introducción
- 2 Sistemas de transición de estados
- 3 Lenguaje de Falluto2.0
- 4 Compilación de Falluto2.0
- 5 Fairness en Falluto2.0
- 6 Interpretación de trazas
- 7 Resultados**
- 8 Ejemplo de uso

A favor

- Lenguaje simple y declarativo para la verificación de sistemas tolerantes a fallas.
- Herramienta que abstrae y automatiza la verificación de manera eficiente.
- Efectividad comprobada en casos reales de verificación (sistema de comunicación inter-satelital).

En contra

- Soporta solo una pequeña porsión de las capacidades de NuSMV.
- La bateria de metapropiedades puede y debería ser ampliada.
- El lenguaje de modelado podría hacerse más flexible.

Tabla de contenidos

- 1 Introducción
- 2 Sistemas de transición de estados
- 3 Lenguaje de Falluto2.0
- 4 Compilación de Falluto2.0
- 5 Fairness en Falluto2.0
- 6 Interpretación de trazas
- 7 Resultados
- 8 Ejemplo de uso**

Ejemplo

Cómo usar Falluto2.0

Fin

FIN