

Apuntes MFIS - Model Checking y Alloy

Model Checking:

❖ ¿Qué es el model checking?

El Model Checking (Comprobación de Modelos) es una técnica automática que, dado un modelo de estados finitos de un sistema, y una propiedad formal, comprueba sistemáticamente si la propiedad se satisface sobre el sistema.

❖ Ventajas

- 1) Es una técnica de verificación general que es aplicable a un amplio rango de aplicaciones.
- 2) Permite la verificación parcial, es decir, las propiedades se pueden verificar individualmente, empezando por la más crítica.
- 3) Es capaz de detectar errores que son improbables que sucedan en una ejecución real.
- 4) Genera contraejemplos que facilitan la depuración del sistema.
- 5) Potencialmente es una tecnología push-button (no requiere mucha interacción con el usuario).
- 6) El interés del model checking está aumentando en la industria.

❖ Desventajas

- 1) Se adapta mejor a sistemas software con mucho código.
- 2) Verifica modelos del sistema, pero no el sistema real.
- 3) No hay garantía de que la verificación sea completa.
- 4) Sufre del problema llamado "Explosión de estados", es decir, no puede soportar altas cantidades de estados.
- 5) Su uso requiere de alguna experiencia en modelado de sistemas.

❖ Grafo de alcanzabilidad

Para aplicar las técnica de verificación formal hay que describir el modelo del sistema como un **grafo**, donde:

- Los nodos son estados.
- Las acciones son flechas con etiquetas.
- El estado inicial es un nodo con una flecha entrante.
- Las trazas son un conjunto de estados que permiten pasar de un estado inicial a uno final.

❖ Definición de canal

Un canal es una estructura de datos intermedia, la cual, permite que los procesos puedan comunicarse.

❖ Bit alternate

También conocido como "*bit de control b*", es utilizado para distinguir las retransmisiones de **m**(mensaje) de las transmisiones de los mensajes siguiente y anterior.

❖ Spin

El **Spin** es un model checker que tiene a **Promela** como lenguaje de entrada. Tiene dos funcionalidades muy relacionadas:

- **Simulador:** permite observar distintas ejecuciones para tener una idea de cómo funciona nuestro modelo.
- **Verificador:** Permite probar propiedades sobre los modelos.
 - Cuando el verificador proporciona un contraejemplo a la propiedad que se está probando, se lanza el simulador para poder observar la traza que ha llevado al error.

❖ Promela

Promela es un lenguaje de modelado orientado al diseño de sistemas software concurrentes. En Promela es fácil modelar los aspectos de comunicación y sincronización de los procesos.

- No es un lenguaje de implementación.
- Los constructores del lenguaje básicos son:
 - Procesos asíncronos;
 - Canales de sincronización y de capacidad finita
 - Instrucciones de sincronización.
 - Datos estructurados
 - No hay Noción del tiempo
 - No hay tipos Reales (Punto Flotante)

❖ Estructuras de kripke




La **estructura de KripKe** es una tupla con 2 componentes:

Tiene en cuenta el modelo de requisitos y un conjunto de restricciones escritas en forma de lógica temporal (Secuencias/árboles de ejecución de un sistema).

❖ LTL (Lógica Temporal)

Es un formalismo para describir “secuencias/árboles” de ejecución de un sistema.

- No se hace medición explícita del tiempo.
- Especifica propiedades que utilizan expresiones como eventualmente o nunca.
- Su semántica hace uso de las estructuras kripke.
- Existen distintas lógicas temporales, por ejemplo.

 En el siguiente estado	
 En algun estado	
 Entodos los estados futuros	
U “until”	
V “release”	
	\wedge Conjunción
	\vee Disjunción
	\neg Negación

Fórmulas de
Camino

Formulas de Estado

Tipos de Propiedades:

- Invariantes: Siempre se cumplen
- De Seguridad: Afirma que bajo ninguna circunstancia el sistema alcanza el estado de error.
 - Las invariantes son de p.de seguridad que se evalúa sobre cada estado alcanzable.
- De viveza: afirma que el sistema siempre está en un estado de progreso. (Siempre habrá algún estado futuro).
- De justicia: se suponen ciertas sobre el sistema que se esta ejecutando. son justas:
 - Justicia incondicional
 - Justicia fuerte
 - Justicia débil

- Dependiendo del grado de exigencia de cumplimiento de la propiedad.

Alloy:

❖ Conjuntos

Es un conjunto es una colección homogénea de objetos distintos:

- **Homogénea** significa que todos los elementos pertenecen a un tipo base o dominio.
- **Distintos** significa que cada elemento aparece a lo sumo una vez en el conjunto.

El conjunto se forma por sus elementos, donde el orden de los mismos es irrelevante.

- Dos conjuntos son iguales si todos los elementos de cada uno son iguales. Es decir, si ambos se componen de los mismos elementos.

Definiciones de conjuntos por:

- Compresión, describiendo las propiedades que tienen los elementos del conjunto.
- Cardinalidad ($\#$), la cual indica el número de elementos que contiene un conjunto.

Un conjunto puede ser subconjunto de otro, así mismo un conjunto puede estar compuesto de distintos subconjuntos.

Entre subconjuntos se pueden utilizar las operaciones de:

- Unión: $A \cup B$
- Intersección: $A \cap B$
- Diferencia: $A - B$

Dos conjuntos son distintos si no tienen ningún elemento en común, por ello a veces será necesario dividir un conjunto en una serie de subconjuntos disjuntos a los que llamaremos particiones.

❖ Dominio e imagen de recorrido

La imagen o Recorrido de una relación binaria es el conjunto de todos sus segundos elementos.

El dominio de una relación binaria es el conjunto de todos sus primeros elementos.

❖ Funciones

Una función f es una relación de *aridad* $n+1$ la cual contiene dos tuplas con los n primeros elementos iguales.

❖ ¿Qué es alloy?

- Es un lenguaje formal basado en lógica de primer orden y soportado por analizador SAT, que permite abstraer el sistema y probar propiedades.

❖ ¿Qué había antes de alloy?

- Los demostradores de teoremas.
- Verificadores de modelos (Model-checking).

❖ Ventajas de alloy

- Una rica estructura de objetos, Clasificación y relaciones
- restricciones en una lógica simple
- Posibilidad de capturar el comportamiento dinámico.
- Capacidad de expresar propiedades en el mismo lenguaje en el que se modelan los sistemas.

❖ Innovaciones de alloy

- Lógica relacional.
- Análisis de pequeño alcance.
- Traducción al analizador SAT.

❖ Usos de alloy

- Sistemas críticos.
- Protocolos de red.
- Seguridad web.
- Verificación de código.

❖ Sintaxis de alloy

Para declarar firmas

Sig Nombre {}

Para declarar una firma abstracta

Abstract Sig Nombre {}

Una firma puede extender de otra

Asimismo puede utilizarse un in

cuando se usa un extend, tiene que ser un solo subconjunto, por otro lado con el in, puede utilizarse varios subconjuntos

Abstract sig Nombre {}

Sig Persona extend Nombre {}

Para generar relaciones

sig Planta {}

Sig Nombre {

techo, suelo: Planta

}

esta es una relación de a uno es decir

Suelo = {(H0, P1)}

Relaciones Unarias:

Nombre = {(N0),(N1)...}

Relaciones Binarias:

direccion={ (N0,D0),(N1,D1)}

Relaciones Ternarias:

direc = {(B0, N0, D0), (B0, N1, D1), (B1, N1, D2)}

La multiplicidad se indica con los operadores

set (de 0 a muchos)
some (de 1 a muchos)
lone (0 o 1)
one (extrictamente 1)

por ejemplo:

```
sig Planta{}  
sig Hombre {  
    techo , suelo: set Planta  
}  
operadores de conjunto:  
    + unión  
    & intersección  
    - Diferencia  
    in subconjunto  
    = Igualdad
```

Restricciones (facts) condiciones que tiene que cumplir el sistema

```
fact {  
    no p: Persona | p in p.^(padre + madre)  
    marido = ~esposa  
}
```

Asertos(Assert o checks) condiciones en el que se comprueba que se cumpla en el sistema (búsqueda de contraejemplos)

```
assert NoSoyMiPadre{  
    no h: Hombre | h.padre = h  
}
```

```
check NoSoyMiPadre  
    fun abuelos(p: Persona): set Persona{  
        p.(madre + padre).padre  
}
```


Predicados(pred) Equivalente a un método en OCL, no tiene ni precondición, ni post-condición. Equivale a un fact, pero se le pueden pasar parámetros.

```
pred miPropioAbuelo(p: Persona){  
  p in abuelos[p]  
}
```