UNIVERSIDAD NACIONAL DE RIO CUARTO

Trabajo de tesis para la obtención del grado de Licenciatura en Ciencias de la Computación

Título : Verificación de modelos con Cálculo-μ

Autor
Luciano Putruele

Director de Tesis: *Dr. Pablo F. Castro*
Co-Director de Tesis: *Dr. Germán Regis*

Rio Cuarto, Argentina
Abril 2016

Resumen

En esta tesis desarrollaremos una herramienta de verificación de modelos (Model Checking), llamada MC2, sobre programas formalizados en un lenguaje lógico simple que también será desarrollado en este trabajo. La herramienta de verificación se encargará de, valga la redundancia, verificar propiedades, caracterizadas en la forma de la lógica temporal Calculo Mu, sobre dichos programa. Cabe destacar que el lenguaje de programación y el verificador funcionan como una sola unidad ya que en un mismo programa MC2 se define tanto el modelo como las propiedades a verificar (si las hay). A todo esto, el metalenguaje utilizado para el desarrollo de estas herramientas es Haskell.

Una motivación para el desarrollo de esta tesis fue la utilización de un lenguaje funcional (Haskell) para el desarrollo de la herramienta en su totalidad, en lugar de utilizar lenguajes imperativos u orientados a objetos.

Lo que se propone con este proyecto es explorar otras alternativas para especificar propiedades que un modelo deba satisfacer, siendo la alternativa en este trabajo el Cálculo Mu, en vez de las lógicas temporales más utilizadas en este tipo de herramientas, como ser LTL, CTL y CTL*, ya que esto trae la ventaja de que el Cálculo Mu es más expresivo que los anteriormente nombrados. Adicionalmente los modelos de los programas MC2 se basan en la definición de reglas de transición usando proposiciones lógicas atómicas con lo cuál es transparente ver la estructura del modelo como una máquina de transición de estados más allá de que internamente se los trata simbólicamente como fórmulas para mejorar el rendimiento.

Palabras clave: .

Agradecimientos

Agradezco a ...

Una dedicatoria muy especial es para ...

List of Figures

Contents

Chapter 1

Introducción

La verificación de modelos o comunmente *model checking* es una técnica automática para verificar sistemas reactivos con una cantidad finita de estados, por ejemplo protocolos de comunicación y diseños de circuitos. Las especificaciones de las propiedades a verificar son expresadas en una lógica temporal proposicional, y el sistema esta modelado como un grafo. Se utiliza una busqueda eficiente para determinar automáticamente si las especificaciones son satisfechas por el grafo. Esta técnica fue desarrollada originalmente en 1981 por Clarke y Emerson. Quielle y Sifakis descubrieron independientemente una técnica similar de verificación poco después. Esta técnica tiene varias ventajas importantes sobre probadores de teoremas para verificación de circuitos y protocolos. La mas importante es que es automática. Normalmente, el usuario provee una representación de alto nivel del modelo y una especificación de la propiedad que se desea verificar. El model checker terminará devolviendo la respuesta True indicando que el modelo satisface la especificación o dará una traza de ejecución a modo de contraejemplo si el modelo no satisface la propiedad. Esta es una propiedad muy importante a la hora de encontrar bugs sutiles.

1.1 Objetivos

El objetivo principal de este proyecto es explorar otras alternativas para especificar propiedades que un modelo deba satisfacer, siendo la alternativa en este trabajo el Cálculo-μ, en lugar de las lógicas temporales más utilizadas en este tipo de herramientas, como ser LTL, CTL y CTL*, además de que esto trae la ventaja de que el Cálculo-μ es más expresivo que los anteriormente nombrados, por lo tanto, las propiedades descritas en LTL,CTL y CTL* pueden ser descritan también usando Cálculo-μ. Como objetivo secundario cabe destacar la utilización del paradigma funcional de programación para el desarrollo de la herramienta en su totalidad, en lugar de utilizar paradigmas imperativos u orientados a objetos que son normalmente mas utilizados en el área.

1.2 Estructura

Primero analizaremos conceptos básicos para la comprensión de esta tesis, conceptos como la verificación de modelos, representación de estos modelos, el concepto de lógicas temporales, y en particular el Cálculo-μ. Más tarde introduciremos la noción de verificación simbólica de modelos para así luego entrar en detalle sobre la implementación del model checker MC2. Luego estableceremos la idea detrás del lenguaje MC2, para entrar luego en detalle con la sintaxis y la semántica del lenguaje. Por último se analizará la implementación concreta del Por último veremos algunos ejemplos para afianzar el entendimiento de la aplicación práctica de esta herramienta.

Chapter 2

Conceptos preliminares

La verificación de modelos o model checking es una técnica automática de verificación de propiedades sobre sistemas con una cantidad finita de estados. Es una alternativa interesante con respecto al testing o las simulaciones ya que a diferencia de estas técnicas, el model checking hace una prueba exhaustiva del sistema, es decir, analiza todas las trazas posibles de la ejecución del sistema en cuestión. Sin embargo, esto trae un problema, esto es el problema de la explosión de estados. Esto ocurre en sistemas con muchas interacciones internas, y que pueden hacer crecer exponencialmente el espacio de estados posibles del sistema, ya que la prueba es exhaustiva no se puede ignorar ningún estado posible. En los últimos años se ha logrado un gran progreso en cómo lidiar con este problema mediante formas más compactas de representar al sistema, como por ejemplo, una representación simbólica.

El proceso del model checking consta de varias tareas:

Modelado: Lo primero es convertir el modelo de un sistema en un formalismo aceptado por la herramienta de verificación de modelos.

Especificación: Es necesario expresar de alguna forma las propiedades que necesitan ser verificadas en el modelo. En general estas propiedades se dan en alguna lógica formal, particularmente lógicas temporales ya que estas pueden expresar comportamientos futuros del sistema.

Verificación: Dado el modelo y la especificación, la tarea de verificar significa explorar exhaustivamente todos los estados posibles del sistema para llegar a la conclusión de que el mismo satisface la especificación o no, en este último caso se suele dar también una traza de error, lo cual ayuda al programador para encontrar fallas en el sistema. Sin embargo, la causa de que el modelo no haya pasado la verificación puede deberse a una especificación incorrecta.

2.1 Modelado de sistemas

En esta sección veremos cómo representar un modelo explícitamente mediante una estructura de Kripke, más tarde veremos otra forma de representación llamada simbólica que representa el modelo mediante una fórmula lógica de primer orden.

Sea AP un conjunto de proposiciones atómicas, una estructura de Kripke M sobre AP es una cuatro-upla $M = (S, S_0, R, L)$ donde
1. S es un conjunto finito de estados.
2. $S_0 \in S$ es el conjunto de estados iniciales.
3. $R \in S \times S$ es una relación de transición total, es decir para cada estado $s \in S$ existe un estado $s' \in S$ tal que $R(s, s')$ vale.
4. $L: S \to 2^{AP}$ es una función que etiqueta a cada estado con el conjunto de proposiciones atómicas que son verdaderas en ese estado. Un camino en la estructura M desde un estado s es una secuencia infinita de estados $p = s_0, s_1, s_2, s_3, ...,$ tal que $s = s_0$ y $R(s_i, s_{i+1})$ vale para todo $i > 0$.

Sea $V = v_1, v_2, ..., v_n$ el conjunto de variables del sistema y sea D el dominio, llamaremos una valuación de V a una función que asocia a cada variable de V un valor de D.

Un estado del sistema se puede representar como una valuación de las variables del sistema. Una proposición atómica de la forma $v = d$ donde $v \in V$ y $d \in D$ será verdadera en un estado s si y solo si $s(v) = d$. Dada una valuación, podemos escribir una fórmula que sea verdadera precisamente para esa valuación, por ejemplo si tenemos $V = \{x, y, z\}$ y la valuación $(x \leftarrow True, y \leftarrow True, z \leftarrow False)$ entonces derivamos la fórmula $(x \wedge y \wedge !z)$. En general, una fórmula puede ser verdadera para varias valuaciones. Si adoptamos la convención de que una fórmula representa el conjunto de todas las valuaciones que la hacen verdadera, entonces podremos describir ciertos conjuntos de estados como fórmulas de primer orden. En particular, el conjunto de los estados iniciales del sistema puede describirse como una fórmula de primer orden S_0 sobre las variables en V. Una transición del sistema se puede representar como un par ordenado de valuaciones, de forma similar podemos describir conjuntos de transiciones mediante una formula para ese par, pero para poder expresar la fórmula se necesita una copia V' de V para hablar del *s*iguiente estado, en V' todas las variables estan primadas. Por ejemplo si tenemos una transición $(x \leftarrow True, y \leftarrow True, z \leftarrow False, (x \leftarrow True, y \leftarrow True, z \leftarrow True))$, podemos derivar la fórmula $(x \wedge y \wedge !z \wedge x' \wedge y' \wedge z')$.

Consideremos el siguiente ejemplo, tenemos $V = \{x, y, z\}$ y $D = \{True, False\}$, $S_0(x, y, z) = (x = True \wedge y = True \wedge z = False)$, y tenemos solo una transición: $z := x \wedge y$, definimos la estructura de kripke de la siguiente manera:

$$S = D \times D \times D$$

$$S_0 = \{(1,1,0)\}$$

$$R = \{((1,1,0),(1,1,1)),((1,1,1),(1,1,1))\}$$

$$L(1,1,0) = \{x = 1, y = 1, z = 0\},$$

$$L(1,1,1) = \{x = 1, y = 1, z = 1\}$$

El único camino posible en esta estructura partiendo del estado inicial es: (1, 1, 0), (1, 1, 1), (1, 1, 1), (1, 1, 1) ...

2.2 Especificación de propiedades

Ahora describiremos una lógica para especificar propiedades deseadas en una estructura de Kripke u otra máquina de transición de estados. La lógica utiliza proposiciones atómicas y operadores como la disyunción y la negación para construir expresiones más complicadas que describan propiedades sobre estados. La lógica temporal es un formalismo que permite describir secuencias de transiciones entre estados en un sistema reactivo, nos interesa saber si en algún momento se llega a un estado determinado o que nunca se llegue a un deadlock. Para esto introduce nuevos operadores especiales que permiten hablar sobre tiempo. Estos operadores pueden combinarse con los operadores lógicos conocidos. Analizaremos a continuación una lógica temporal muy potente llamada Cálculo-μ.

2.3 Cálculo-μ

Sea $M = (S, T, L)$ una estructura de Kripke y sea $VAR = Q, Q1, Q2, \dots$ un conjunto de variables relacionales, donde a cada variable relacional se le puede asignar un subconjunto de S, construimos una μ-fórmula como sigue:

-Si p pertenece a AP entonces p es una fórmula. -Si Q pertenece a VAR, Q es una fórmula. -Si f y g son fórmulas, entonces $\neg f$, $f \vee g$, y $f \wedge g$ son fórmulas. -Si f es una fórmula, entonces $\Box f$ y $\Diamond f$ son fórmulas. -Si $Q \in VAR$ y f es una fórmula entonces $\mu Q.f$ y $\nu Q.f$ son fórmulas

Las variables pueden estar libres o ligadas en una fórmula a través de un operador de punto fijo. Una fórmula cerrada es una fórmula sin variables libres.

El significado intuitivo de $\diamond f$ es "Es posible realizar una transición a un estado donde f vale", similarmente $\square f$ significa "f vale en todos los estados alcanzables por medio de una transición" Los operadores μ y ν expresan puntos fijos menores y mayores respectivamente. El conjunto vacío de estados se denota con $False$ y el conjunto de todos los estados S se denota con $True$.

Ejemplos

-$\nu Z \cdot f \wedge \square Z$ se interpreta como "f es verdadera siempre en todo camino" -$\mu Z \cdot f \vee \diamond Z$ se interpreta como "existe un camino hacia un estado donde f vale" -$\nu Z \cdot \diamond T \wedge \square Z$ se interpreta como "no hay estados que no tengan transiciones hacia otros estados"

Formalmente, una fórmula f se interpreta como un conjunto de estados donde f es verdadera, escribimos este conjunto como $[[f]]$ sobre un sistema de transición de estados M y un ambiente $e : VAR \rightarrow 2^S$, denotaremos $e[Q \leftarrow W]$ como un ambiente que es igual a e solo que Q ahora tiene el valor W. el conjunto $[[f]]$ sobre M y e se define recursivamente de la siguiente manera:

$$[[p]]Me = \{s \mid p \in aL(s)\}$$

$$[[Q]]Me = e(Q)$$

$$[[\neg f]]Me = S \setminus [[f]]Me$$

$$[[f \wedge g]]Me = [[f]]Me \cap [[g]]Me$$

$$[[f \vee g]]Me = [[f]]Me \cup [[g]]Me$$

$$[[\diamond f]]Me = \{s \mid \exists t : s \rightarrow t \wedge t \in [[f]]Me\}$$

$$[[\square f]]Me = \{s \mid \forall t : s \rightarrow t \rightarrow t \in [[f]]Me\}$$

$[[[\mu Q f]]Me$ es el menor punto fijo del predicado transformador $t : 2^S \rightarrow 2^S$ definido como $t(W) = [[f]]Me[Q \leftarrow W]$ $[[\nu Q f]]Me$ es el mayor punto fijo del predicado transformador $t : 2^S \rightarrow 2^S$ definido como $t(W) = [[f]]Me[Q \leftarrow W]$