Introducción a la Verificación Formal
IIC3800

Pablo Barceló

Motivación

Los sistemas de hardware y software son cada vez más grandes y complejos.

- El número de transistores el año 1979 en procesador 8088 era de 29,000 a 3 micrones.
- En Pentium-4 del año 2000 era de 42 millones a 0.18 micrones.
- En procesador Dual-core del año 2005 es de 290 millones a 0.065 micrones.

Además, dependemos de nuestros sistemas computacionales más que nunca antes.

Motivación

Por tanto, es cada vez más fácil que se produzcan fallas en las aplicaciones, y estas fallas por su parte se vuelven cada vez más difíciles de detectar.

Problema! En ciertas aplicaciones no pueden tolerarse las fallas:

- Diagnóstico y aparataje médico, misiones espaciales, etc.

Además, en ciertas aplicaciones mientras antes se detecte una falla más barato es para el productor.

- Por ejemplo, a Intel el problema de punto-flotante en procesador Pentium le costó 500 millones de dólares.

Fallas en sistemas computacionales críticos: Therac 25

"A computer-controlled radiation therapy machine called the Therac-25 made by Atomic Energy of Canada overdosed six people between June, 1985 and January 1987... In some cases the only indication that something was wrong was the cryptic message:

malfunction 54

The error was a timing problem on data entry."

Wikipedia

Fallas en sistemas computacionales críticos: Therac 25

"The Therac-25 could deliver radiation as either a beam of electrons or a beam of X-rays. If the operator entered 'x' for x rays, the setting of the magnets took 8 seconds. If the operator discovered she had made a mistake and fixed the entry to be 'e' within that 8 seconds, even though the screen refleected the change, the change did not affect the remaining program."

Wikipedia

Fallas en sistemas computacionales con costo monetario

"The floating point processor of Intel Pentium chips manufactured before a certain date had a bug in the FDIV operation. Approximately 2 million chips had the flaw. The FDIV operation didn't always give precise results. According to sources on the web, the bug could be seen by entering the following formula in the Windows calculator:

$$(4195835/3145727) \times 3145727 - 4195835)$$

It produced 512 instead of 0."

Wikipedia

Fallas en sistemas computacionales con costo monetario

EE Times had an article on it in Nov, 1994:

> *"This was a very rare condition that happened once every 9 to 10 billion operand pairs", said Steve Smith, a Pentium engineering manager at Intel. Smith emphasized that the anomaly would not affect the average user. Speaking of Nicely, Smith said: "He's the most extreme user. He spends round-the-clock time calculating reciprocals. What he observed after running this for months is an instance where we have eight decimal points correct, and the ninth not showing up correctly. So you get an error in the ninth decimal digit to the right of the mantissa. I think even if you're an engineer, you're not going to see this."*
>
> EE Times, November 7, 1994, Issue 822, page 1

Fallas en sistemas computacionales con costo monetario

¿Qué sucede después?

- Intel ofrece reemplazar el procesador basado en la "necesidad" del cliente.
- Caen las acciones de Intel, gran preocupación.
- Diciembre 1994: Intel ofrece cambiar el procesador a quien lo desee.
- Costo estimado a Intel: 500 millones de dólares.
- Intel contrata un grupo de expertos para realizar en el futuro verificación formal de sus procesadores.

Seguridad de sistemas computacionales

"Smart cards, the size of a credit card, have a microprocessor and memory, along with a mini operating system. They can run multiple applications, which may be downloaded after the card is in use. (This feature is currently disabled.) These applets can carry out various functions such as being an electronic wallet, carry health information, etc.
Because of the high security considerations, a European project is attempting to verify the code and operating system of these cards for non-interference between applications. For these companies security is their product. "

verificard.org

Motivación

Pregunta: ¿Cómo podemos asegurarnos que un sistema computacional hace lo que queremos que haga, y no hace lo que no queremos que haga?

Chequear la corrección de un producto es más lento y costoso que diseñarlo.

- Por tanto, es deseable automatizar al máximo este proceso.

Verificación formal

El área de Verificación formal estudia los fundamentos teóricos y la implementación de técnicas de verificación de sistemas computacionales.

- Ocupa técnicas matemáticas para asegurar la validez de un sistema con respecto a una especificación.
- Es uno de los ejemplos más exitosos de razonamiento automático aplicado a la computación (ganó el Premio Kanellakis de la ACM en 1998 por tal razón).
- Estas técnicas son cada vez más aplicadas en la industria: IBM, Lucent, AT&T, Motorola, etc.

Verificación formal

La verificación formal consta de tres etapas:

- Modelación: Se construye un modelo matemático de los posibles comportamientos del sistema.
- Especificación: Se especifica en un lenguaje formal los comportamientos deseables del sistema.
- Verificación: Se chequea si el modelo satisface la especificación.

Es formal porque el modelo y la especificación son objetos matemáticos. Es verificación porque el análisis responde con certeza si la especificación se cumple o no (otros tipos de verificación incluyen *testeo*, *simulación*, etc., que no son exhaustivos).

Verificación automatizada

Verificación automatizada: La verificación de la especificación en el modelo se realiza de forma algorítmica.

- La idea es simular exhaustivamente todos los posibles comportamientos del sistema en busca de fallas.

Este tipo de verificación, comúnmente llamado model checking, es el que estudiaremos en el curso. (Propuesto por Clarke y Emerson el año 1981).

No siempre la verificación automatizada es posible (¿Por qué?).

Verificación interactiva

Otro tipo de verificación es la verificación interactiva. En esta el proceso de verificación consiste en probar un teorema desde un conjunto de axiomas.

- Este es el método más general que existe.
- Puede utilizar la ayuda de un demostrador automático de teoremas.
- Al contrario de la verificación automatizada, sólo puede ser realizada por expertos en lógica.

Más razones para estudiar model checking

Es un método general con aplicaciones en verificación de hardware, verificación de sistemas de software, verificación de protocolos criptográficos, etc.

Hay un creciente interés de la industria por aplicar este tipo de técnicas.

Posee una teoría con fundamentos matemáticos interesantes:

- Lógicas temporales.
- Teoría de autómatas.
- Algoritmos y estructuras de datos.

Sistemas reactivos

Estudiaremos un tipo general de sistemas computacionales llamados sistemas reactivos:

- Interactúan con su ambiente aceptando peticiones y produciendo respuestas.

Ejemplo: Una red de computadores, un servicio web, un sistema operativo, un microprocesador, etc., pueden ser vistos como sistemas reactivos.

Nótese que ciertas computaciones en un sistema reactivo pudieran no terminar.

Más sobre: Modelación

Algunas propiedades de la modelación:

- Describe la interacción y el control del sistema. No intenta describir manipulación de datos ni estructuras de datos complejas.
- El lenguaje de modelación debe incluir no-determinismo:
  - Si un mensaje llega puede traspasarse o ser eliminado.
- Un sistema puede ser modelado a diferentes niveles de abstracción (y, por tanto, tener muchos diferentes modelos).
- Formas de describir modelos incluyen autómatas, extensiones de estos para tratar variables, autómatas equipados con mecanismos de comunicación, etc.

Más sobre: Especificación

La especificación se refiere a los requerimientos que queremos que nuestro sistema computacional cumpla.

Algunos tipos de especificación:

- Pólizas de seguridad: Algo malo nunca ocurrirá (La temperatura no excederá los 100C, el semáforo no estará siempre en verde, etc.).
- Pólizas de vivacidad: Algo bueno ocurrirá eventualmente (Alguna vez lloverá, el semáforo pasará a amarillo, etc.).

Más sobre: Especificación y lógicas temporales

En model checking los requerimientos se especifican comúnmente en lógica temporal:

- Propuesto por Pnueli en 1977, la idea le dió un premio Turing.
- Estas lógicas se evalúan en secuencias de asignaciones a las variables del sistema.
- Combinan buenas propiedades de expresividad y complejidad.

Más sobre: Especificación y autómatas

La especificación también puede hacerse utilizando un autómata:

- El modelo se ve como un generador de un lenguaje formal.
- La especificación es un lenguaje formal dado.
- La verificación consiste en resolver un problema de inclusión entre lenguajes formales.

Este método está basado en la teoría de autómatas sobre palabras infinitas. Fue desarrollado por Vardi y Wolper el año 1984.

Más sobre: Model checking

Inicialmente creado para sistemas finitos y una lógica temporal particular (CTL). Actualmente aplica a un escenario mucho más general.

Mayor problema: El número de estados del sistema crece exponencialmente con el número de variables (explosión del espacio de estados).

Una gran parte del área de model checking intenta buscar métodos que resuelvan parcialmente este problema.

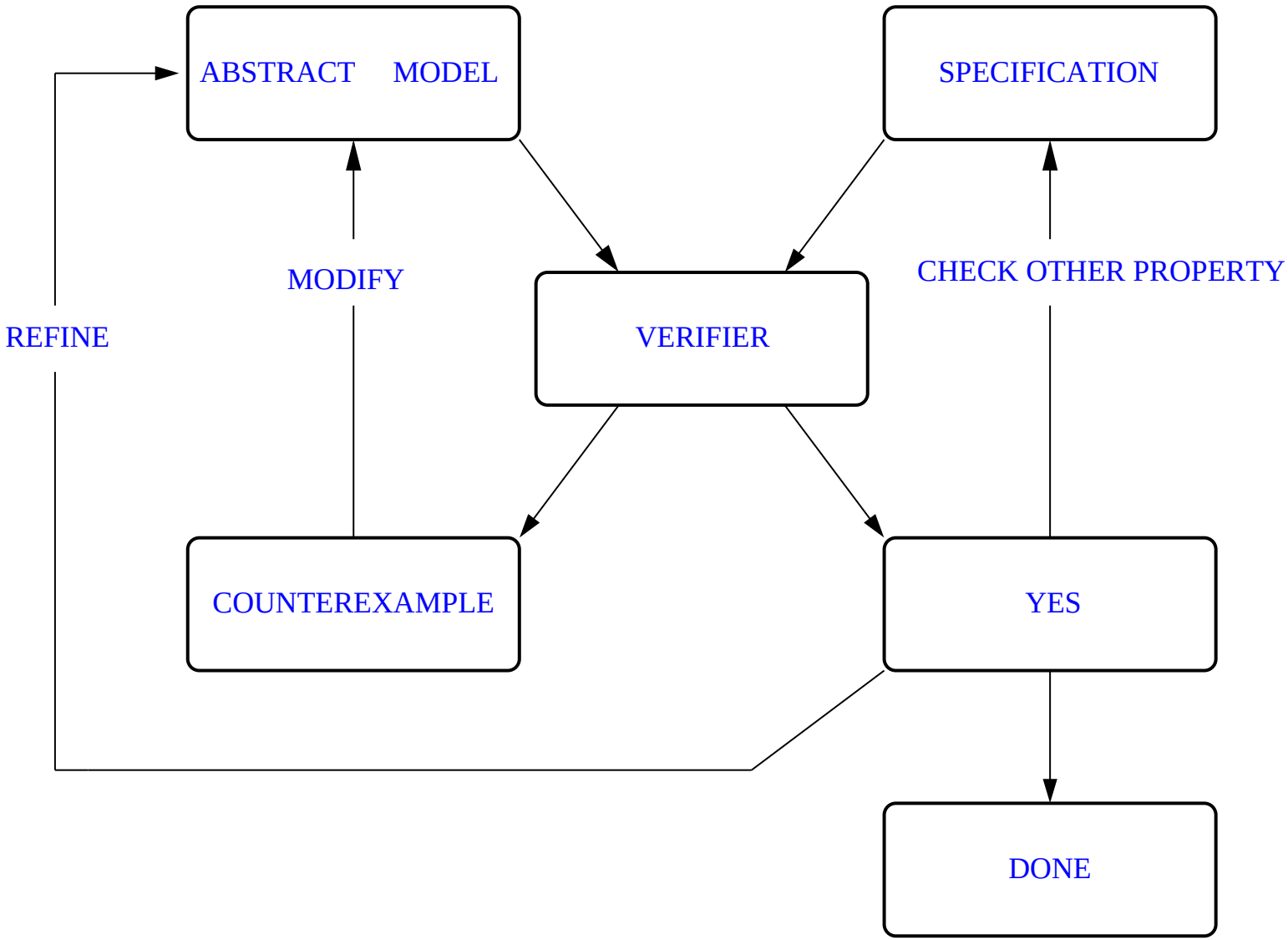
Más sobre: Model checking

El model checking responde con certeza si la especificación es cierta o no en el sistema.

En muchos casos cuando la respuesta es negativa, proporciona además un contraejemplo o diagnóstico.

Esta información es muy útil para solucionar los problemas del sistema.

Más sobre: Model checking y verificación jerárquica



Verificación de hardware

El model checking ha sido particularmente exitoso en la verificación de hardware (e.g. diseño de microprocesadores en Lucent, IBM, Intel, Motorola y Siemens).

Existen además herramientas de verificación ya en el mercado (e.g. *FormalCheck* de Lucent).

Este éxito se explica porque el model checking calza bien con las metodologías de diseño existentes (los diseñadores normalmente ocupan lenguajes de descripción de alto nivel como VHDL y Verilog, además de métodos de verificación para éstos).

Verificación de software

Diseño de alto nivel no es muy común en software, y, por tanto, el model checking no ha logrado gran impacto aún en la comunidad.

En algunos sistemas que son "safety-critical", como la aviación, sí ha podido ser implementado.

La popularidad creciente de lenguajes síncronos como Esterel permite tener esperanzas a futuro.

Limitaciones

El dominio de aplicación actual son sistemas de control e interacción: Poco flexible aún para tratar bases de datos, procesadores de texto, etc.

El model checking trabaja sobre un *modelo* del sistema:

- Su fuerza radica en detectar errores, no chequear corrección.

El mismo problema pero ahora con respecto a la especificación.

Aunque la tecnología avance, los problemas de complejidad computacional y explosión de espacio de estados seguirán siendo insalvables en muchos casos.