UNIVERSIDAD DE GRANADA

Departamento de Ciencias de la Computación
e Inteligencia Artificial

Modelos Avanzados de Computación

Práctica 1

El problema de la parada

Curso 2014-2015

Doble Grado en Ingeniería Informática y Matemáticas

Práctica 1

El problema de la parada

1. Muestre que, si pudiésemos resolver el problema de la parada, también podríamos resolver muchos problemas matemáticos no resueltos. Ilustre cómo se podría demostrar la conjetura de Goldbach si tuviésemos un algoritmo que resuelva el problema de la parada.

   > Conjetura de Goldbach: Todo número par mayor que 2 puede descomponerse como la suma de dos números primos.

2. A partir de los resultados de Turing sobre el problema de la parada, demuestre el Teorema de Incompletitud de Gödel: Dado cualquier conjunto consistente y computable de axiomas, existe una sentencia verdadera sobre números enteros que no puede demostrarse a partir de dichos axiomas. Consistente hace referencia a que no se puede derivar ninguna contradicción, mientras que computable quiere decir que existe un número finito de axiomas o bien, aunque haya un número infinito, existe un algoritmo que nos permite generarlos (si no incluyésemos este requisito, podríamos partir de un conjunto de axiomas formado por todas las sentencias verdaderas sobre números enteros; lo cual, de todas formas no sería demasiado útil en la práctica).

3. A partir de los resultados de Turing sobre el problema de la parada, demuestre el Segundo Teorema de Incompletitud de Gódel: Si una teoría F es consistente, entonces F no puede demostrar su propia consistencia. En otras palabras, las únicas teorías matemáticas que pueden demostrar su consistencia son precisamente aquéllas que no son consistentes.

4. Demuestre, por reducción al problema de la parada, que no es posible desarrollar un depurador automático que sea capaz de, en general:

   - Determinar si un programa incluye algún bucle infinito.
   - Determinar si un programa mostrará por pantalla un mensaje determinado.
   - Determinar si un programa llegará a ejecutar una sentencia determinada.
   - Determinar si una variable se inicializará siempre antes de ser utilizada.

Esto no quiere decir que no se puedan construir sistemas en los que sea posible realizar demostraciones sobre el comportamiento de un programa, sólo que no existen soluciones universales a tales problemas. Por ejemplo, la máquina virtual de Java está diseñada para poder permitir determinados tipos de demostraciones y un compilador de Java es capaz de detectar muchos de estos errores en tiempo de compilación. Busque información en Internet sobre el tema.

5. Muestre que no es posible determinar, en general, si dos programas hacen lo mismo. Esto es, dados dos programas P y Q, demuestre que no es decidible el problema de determinar si ambos calculan la misma función: P(x) = Q(x) para todo x.

   Observe que esto implica que, en general, no se puede determinar si un programa original P y su versión optimizada P' producida por un compilador calculan exactamente lo mismo.

6. Usando el resultado del ejercicio anterior, muestre cómo siempre se puede derrotar a un troyano o caballo de Troya que intente pasar desapercibido (un troyano es un software malicioso que se presenta al usuario como un programa aparentemente legítimo e inofensivo, pero que, al ejecutarlo, le brinda a un atacante acceso remoto al equipo infectado mediante la creación de una puerta trasera o backdoor).

   Ken Thompson ('Reflections on Trusting Trust', 1984, `http://cm.bell-labs.com/who/ken/trust.html`) defendía que podía colocar un troyano en un compilador de C que compilaría mal el login para permitirle acceder a cualquier sistema Unix que se compilase con su compilador de C y que, además, podía compilar mal el compilador de C para insertar una copia del troyano sin que el troyano en sí apareciese en el código fuente del compilador de C. Demuestre que estaba equivocado.

7. Demuestre que no existe ningún algoritmo que sea capaz de determinar si un problema matemático es decidible o no (esto es, un programa que sea capaz de indicarnos si una sentencia es decidible, sin llegar determinar si es cierta o falsa).

   En la práctica, esto quiere decir que, incluso aunque dispusiésemos de recursos ilimitados (CPU y memoria), siempre habrá cosas que no podremos demostrar formalmente con la ayuda de un sistema de demostración de teoremas. Pese a ello, los sistemas de verificación formal de programas siguen siendo útiles para demostrar las propiedades formales de determinados programas, siempre que impongamos las restricciones necesarias que nos permitan realizar tales demostraciones de forma eficiente (en ámbitos bien delimitados, no de forma universal).

8. Muestre cómo se podría resolver el problema de la parada cuando imponemos una restricción sobre la cantidad de memoria que puede utilizar un programa. Obviamente, necesitará un sistema que disponga de muchísima más memoria de la que permitimos que utilice el programa...