MODELOS DE COMPUTACIÓN Y COMPLEJIDAD

Grado en Ingeniería Informática. Tecnologías Informáticas

ETS Ingeniería Informática. Universidad de Sevilla (Curso 2021-2022)

Problemas de FUNCIONES GOTO-COMPUTABLES

EJERCICIO 19.

Sea $\texttt{GOTO}_f$ el modelo de computación cuyos procedimientos mecánicos son los programas GOTO en los que **toda instrucción condicional** de dicho programa deben ser del tipo $I \equiv \texttt{IF } V \neq 0 \texttt{ then GOTO } L$, siendo L una etiqueta que sólo puede aparecer a la izquierda de intrucciones del programa que sean "posteriores" a I. Probar que:

1. Toda función $\texttt{GOTO}_f$-computable es **total**.
2. Los modelos de computación $\texttt{GOTO}_f$ y $\texttt{GOTO}_l$ **no son equivalentes**. Específicamente, probar que existen funciones computables en el modelo $\texttt{GOTO}_f$ que, en cambio, no lo son en el modelo $\texttt{GOTO}_l$.
3. Toda función $\texttt{GOTO}_f$-computable de aridad 1 tiene **rango finito**.
4. El modelo de computación GOTO es **estrictamente más potente** (es decir, *calcula* más funciones) que el modelo de computación $\texttt{GOTO}_f$.

SOLUCIÓN:

1. Para demostrar que toda función $\texttt{GOTO}_f$-computable es total, será suficiente establecer que todo programa del modelo de computación $\texttt{GOTO}_f$ **para** sobre cualquier dato de entrada. Con el fin de simplicar la notación, vamos a probar el resultado para funciones 1-arias, si bien el resultado se extiende de manera natural a funciones de aridad $k \geq 2$, simplemente sustituyendo los números naturales por k-tuplas de números naturales.

   En realidad ("*matando dos pájaros de un tiro*"), vamos a demostrar un poco más: específicamente, vamos a probar que todo programa del modelo de computación $\texttt{GOTO}_f$ **para sobre cualquier dato de entrada** y, además, devuelve un valor que es **menor o igual que la longitud** de dicho programa.

   Para ello, se considera la fórmula $\theta(k)$ definida para cada $k \in \mathbb{N}$, como sigue:

   $$\theta(k) \equiv [\forall P \ (P \text{ programa } \texttt{GOTO}_f \ \wedge |P| = k \rightarrow (\forall x \in \mathbb{N} \ (P(x)\downarrow \wedge \ P(x) \leq k)))]$$

   Se ha de probar que la fórmula $\theta(k)$ es verdadera para cada número natura k. Probémoslo por inducción sobre k.

   - **Caso base:** $k = 0$.

     Si P es un programa $\texttt{GOTO}_f$ cuya longitud es 0 entonces P es el programa vacío. Por tanto, se verifica que $\forall x \in \mathbb{N} \ (P(x)\downarrow \wedge \ P(x) = 0 \leq 0)$.

– **Paso inductivo: Sea $k \in \mathbb{N}$ tal que la fórmula $\theta(k)$ es verdadera. Veamos que la fórmula $\theta(k+1)$ también es verdadera**.

Para ello, sea P un programa $\texttt{GOTO}_f$ cuya longitud es $k+1$. Sea I_{k+1} la última instrucción de P y sea Q el programa obtenido a partir de P eliminando la instrucción I_{k+1}. Entonces, Q es un programa $\texttt{GOTO}_f$ cuya longitud es k. Teniendo presente que la fórmula $\theta(k)$ es verdadera, se tendrá que $\forall x \in \mathbb{N}\ (Q(x)\downarrow \wedge\ Q(x) \leq k)$. En tales circunstancias, hemos de probar que $\forall x \in \mathbb{N}\ (P(x)\downarrow \wedge\ P(x) \leq k+1)$. Y lo vamos a hacer teniendo presente que la ejecución de P se obtiene ejecutando $Q(x)$ y, posteriormente, ejecutando la instrucción I_{k+1} sobre la configuración de parada $\sigma_h(x)$ de $Q(x)$.

* Si I_{k+1} no es una instrucción condicional, entonces tras ejecutar esa instrucción sobre la configuración $\sigma_h(x)$, el programa P **para** y, además, $P(x) \leq Q(x)+1 \leq k+1$.
* Si I_{k+1} es una instrucción condicional del tipo $\texttt{IF}\ V \neq 0\ \texttt{then GOTO}\ L$, por la definición sintáctica del modelo de computación $\texttt{GOTO}_f$, L no puede etiquetar a la izquierda a **ninguna** instrucción de P. Por tanto, tras ejecutar I_{k+1} sobre la configuración $\sigma_h(x)$, el programa P **para** y, además, $P(x) = Q(x) \leq k < k+1$.

De lo que antecede se deduce que la fórmula $\theta(k+1)$ es verdadera.

Esto que completa la demostración de que $\forall k \in \mathbb{N}\ (\theta(k))$.

2. Consideremos la función total $f : \mathbb{N} \to \mathbb{N}$ definida como sigue: $f(0) = 1$ y $f(x) = 0$, para cada número natural $x > 0$. Entonces se tiene lo siguiente:

   – Por una parte, el siguiente programa

   $$P \equiv \begin{cases} \texttt{IF}\ X \neq 0\ \texttt{GOTO}\ E \\ \texttt{Y} \leftarrow \texttt{Y} + \texttt{1} \end{cases}$$

   verifica que $P(0) = 1$ y $P(x) = 0$, para cada número natural $x > 0$. Es decir, P es un programa $\texttt{GOTO}_f$ que calcula la función total f.

   – Por otra, teniendo presente que la función f **no es constante** resulta (ver el apartado 3 del ejercicio 18) resulta que f **no es computable** en el modelo $\texttt{GOTO}_l$.

   En consecuencia, existen funciones que son computables en el modelo $\texttt{GOTO}_f$ y, en cambio, no lo son en el modelo $\texttt{GOTO}_l$ (por ejemplo, la función f antes citada).

   Por tanto, los modelos de computación $\texttt{GOTO}_f$ y, $\texttt{GOTO}_l$ **no son equivalentes**. Más aún, teniendo presente que todo programa del modelo $\texttt{GOTO}_l$ es un programa del modelo $\texttt{GOTO}_f$ concluimos que el modelo de computación $\texttt{GOTO}_f$ es **estrictamente más potente** que el modelo de computación $\texttt{GOTO}_l$.

3. Sea $f : \mathbb{N} \to \mathbb{N}$ una función total de aridad 1 que es computable en el modelo $\texttt{GOTO}_f$. Sea P un programa $\texttt{GOTO}_f$ que calcula la función f. Si la longitud de P es k entonces, de la prueba del apartado 1, se deduce que $P(x) \leq k$, para cada $x \in \mathbb{N}$. Por tanto, se tendrá que $f(x) \leq k$, para cada $x \in \mathbb{N}$ y, en consecuencia, el rango de la función total f será un conjunto finito (específicamente, el rango de f estará contenido en el conjunto de números naturales $\{0, \ldots, k\}$).

4. En primer lugar, obsérvese que todo programa $\texttt{GOTO}_f$ es, en particular, un programa GOTO. Por tanto, toda función $\texttt{GOTO}_f$-computable será, así mismo, GOTO .

   Ahora bien, la función vacía es, obviamente, GOTO pero, en cambio, **no** es $\texttt{GOTO}_f$-computable ya que **no** es una función **total**. Es decir, existen funciones GOTO que **no** son $\texttt{GOTO}_f$-computables.

   En consecuencia, el modelo de computación GOTO es **estríctamente más potente** que el modelo de computación $\texttt{GOTO}_f$.

– –