Programas y Máquinas I

José de Jesús Lavalle Martínez

Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación
Computabilidad CCOS 257

Contenido

Motivación

- En este capítulo nos centramos en otra forma de formalizar el concepto de calculabilidad efectiva: los programas de máquinas de registros.

Motivación

- En este capítulo nos centramos en otra forma de formalizar el concepto de calculabilidad efectiva: los programas de máquinas de registros.
- Nuestro primer objetivo es mostrar que todas las funciones parciales recursivas generales también son computables mediante máquinas de registros.

Motivación

- En este capítulo nos centramos en otra forma de formalizar el concepto de calculabilidad efectiva: los programas de máquinas de registros.
- Nuestro primer objetivo es mostrar que todas las funciones parciales recursivas generales también son computables mediante máquinas de registros.
- Este hecho nos permitirá aplicar nuestro trabajo en el Capítulo 2 para ver que muchas funciones cotidianas son computables por máquinas de registros.

Motivación

- En este capítulo nos centramos en otra forma de formalizar el concepto de calculabilidad efectiva: los programas de máquinas de registros.
- Nuestro primer objetivo es mostrar que todas las funciones parciales recursivas generales también son computables mediante máquinas de registros.
- Este hecho nos permitirá aplicar nuestro trabajo en el Capítulo 2 para ver que muchas funciones cotidianas son computables por máquinas de registros.
- Usando esto, podremos construir un programa universal, es decir, un programa para calcular la función parcial $\Phi(w, x) =$ el resultado de aplicar el programa codificado por w a la entrada x.

Máquinas de registros I

Recuerde del Capítulo 1 que un programa de máquina de registros es una secuencia finita de instrucciones de los siguientes tipos:

- "Incrementa r", $I\ r$ (donde $0 \leqslant r \leqslant K$): El efecto de esta instrucción es aumentar el contenido del registro r en 1. Luego, la máquina pasa a la siguiente instrucción del programa (si corresponde).

Máquinas de registros I

Recuerde del Capítulo 1 que un programa de máquina de registros es una secuencia finita de instrucciones de los siguientes tipos:

- "Incrementa r", $I\ r$ (donde $0 \leqslant r \leqslant K$): El efecto de esta instrucción es aumentar el contenido del registro r en 1. Luego, la máquina pasa a la siguiente instrucción del programa (si corresponde).
- "Decrementa r", $D\ r$ (donde $0 \leqslant r \leqslant K$): El efecto de esta instrucción depende del contenido del registro r. Si ese número es distinto de cero, se decrementa en 1 y la máquina no pasa a la siguiente instrucción, sino a la siguiente de la siguiente. Pero si el número en el registro r es cero, la máquina simplemente pasa a la siguiente instrucción. En resumen, la máquina intenta disminuir el registro r y, si tiene éxito, se salta una instrucción.

Máquinas de registros I

Recuerde del Capítulo 1 que un programa de máquina de registros es una secuencia finita de instrucciones de los siguientes tipos:

- "Incrementa r", $I\ r$ (donde $0 \leqslant r \leqslant K$): El efecto de esta instrucción es aumentar el contenido del registro r en 1. Luego, la máquina pasa a la siguiente instrucción del programa (si corresponde).
- "Decrementa r", $D\ r$ (donde $0 \leqslant r \leqslant K$): El efecto de esta instrucción depende del contenido del registro r. Si ese número es distinto de cero, se decrementa en 1 y la máquina no pasa a la siguiente instrucción, sino a la siguiente de la siguiente. Pero si el número en el registro r es cero, la máquina simplemente pasa a la siguiente instrucción. En resumen, la máquina intenta disminuir el registro r y, si tiene éxito, se salta una instrucción.
- "Jump q", $J\ q$ (donde q es un número entero: positivo, negativo o cero): todos los registros se dejan sin cambios. La máquina toma como siguiente instrucción la $q-$ésima instrucción que sigue a ésta en el programa (si $q \geqslant 0$), o la $|q|-$ésima instrucción que precede a ésta (si $q < 0$). La máquina se detiene si no existe tal instrucción en el programa. La instrucción $J\ 0$ da como resultado un bucle, en el que la máquina ejecuta esta instrucción una y otra vez.

Máquinas de registros II

- Ahora supongamos que f es una función parcial de aridad n sobre $\mathbb{N}$.

Máquinas de registros II

- Posiblemente habrá un programa $\mathcal{P}$ tal que si iniciamos una máquina de registros (que tiene todos los registros a los que $\mathcal{P}$ se refiere) con $x_1, \ldots, x_n$ en los registros $1, \ldots, n$ y 0 en los demás registros, y aplicamos el programa $\mathcal{P}$, entonces se cumplen las siguientes condiciones:

Máquinas de registros II

- Posiblemente habrá un programa $\mathcal{P}$ tal que si iniciamos una máquina de registros (que tiene todos los registros a los que $\mathcal{P}$ se refiere) con $x_1, \ldots, x_n$ en los registros $1, \ldots, n$ y 0 en los demás registros, y aplicamos el programa $\mathcal{P}$, entonces se cumplen las siguientes condiciones:
  - Si $f(x_1, \ldots, x_n)$ está definida, entonces el cálculo finalmente termina con $f(x_1, \ldots, x_n)$ en el registro 0. Además, el cálculo termina buscando la $(p+1)-$ésima instrucción, donde p es la longitud de $\mathcal{P}$.

Máquinas de registros II

- Posiblemente habrá un programa $\mathcal{P}$ tal que si iniciamos una máquina de registros (que tiene todos los registros a los que $\mathcal{P}$ se refiere) con $x_1, \ldots, x_n$ en los registros $1, \ldots, n$ y 0 en los demás registros, y aplicamos el programa $\mathcal{P}$, entonces se cumplen las siguientes condiciones:
  - Si $f(x_1, \ldots, x_n)$ está definida, entonces el cálculo finalmente termina con $f(x_1, \ldots, x_n)$ en el registro 0. Además, el cálculo termina buscando la $(p+1)-$ésima instrucción, donde p es la longitud de $\mathcal{P}$.
  - Si $f(x_1, \ldots, x_n)$ no está definida, entonces el cálculo nunca termina.

Máquinas de registros III

- Si existe tal programa $\mathcal{P}$, decimos que $\mathcal{P}$ calcula f, observe que cuando iniciamos la ejecución de un programa, hay tres posibilidades:

Máquinas de registros III

- podría ejecutarse para siempre;

Máquinas de registros III

(II) podría detenerse “bien”, al buscar la primera instrucción que no está allí;

Máquinas de registros III

III podría detenerse “mal”, ya sea intentando saltar a algún lugar antes del inicio del programa o intentando avanzar a una instrucción, inexistente, después de la última instrucción del programa.

Máquinas de registros III

- En nuestra definición de "$\mathcal{P}$ calcula f", hemos elegido descartar paradas incorrectas.

Máquinas de registros III

- Esto será conveniente para ejecutar programas de principio a fin.

Máquinas de registros IV

- Por ejemplo, la función de suma $x + y$ se calcula mediante un programa de máquina de registros dado en el Capítulo 1.

Máquinas de registros IV

- Por ejemplo, la función de suma $x + y$ se calcula mediante un programa de máquina de registros dado en el Capítulo 1.
- Además, en el Capítulo 1, vimos algunas subrutinas básicas:

CLEAR r	Borra el registro r.
MOVE from r to s	Se borra el registro r.
COPY from r to s using t	El registro r no cambia.

Máquinas de registros IV

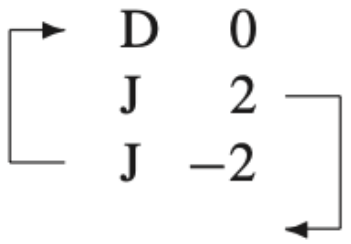
- Por ejemplo, la función de suma $x + y$ se calcula mediante un programa de máquina de registros dado en el Capítulo 1.
- Además, en el Capítulo 1, vimos algunas subrutinas básicas:

CLEAR r	Borra el registro r.
MOVE from r to s	Se borra el registro r.
COPY from r to s using t	El registro r no cambia.

- Estas subrutinas tienen longitud 3, 7 y 15, respectivamente.

Máquinas de registros V

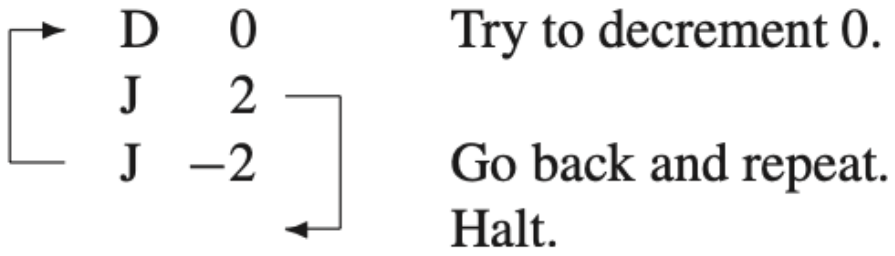
- Para un ejemplo trivial, la función constante cero de aridad n se calcula mediante el programa vacío y por nuestro programa de tres líneas para borrar el registro 0:



```
D  0    Try to decrement 0.
J  2
J  −2   Go back and repeat.
        Halt.
```

Máquinas de registros V

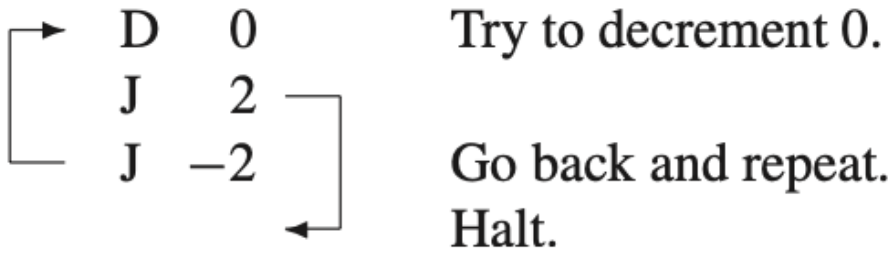
- Para un ejemplo trivial, la función constante cero de aridad n se calcula mediante el programa vacío y por nuestro programa de tres líneas para borrar el registro 0:



- Las flechas son para ayudarnos a ver lo que hace el programa.

Máquinas de registros V

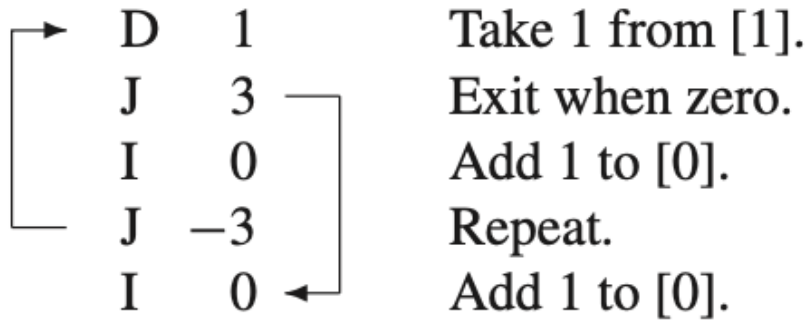
- Para un ejemplo trivial, la función constante cero de aridad n se calcula mediante el programa vacío y por nuestro programa de tres líneas para borrar el registro 0:



- Las flechas son para ayudarnos a ver lo que hace el programa.
- Además, al agregar algunos instrucciones de incremento, podemos ver que cualquier función constante total se calcula mediante algún programa de registros.

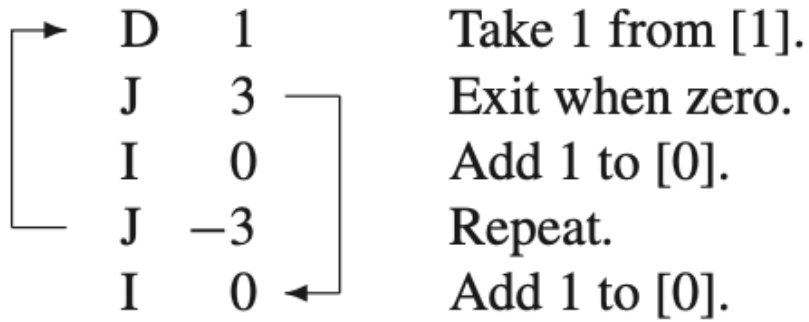
Máquinas de registros VI

- La función sucesora de aridad uno $S(x) = x + 1$ es calculada por el programa que mueve el contenido del registro 1 (llame a este número [1]) al registro 0 y luego incrementa el número:



Máquinas de registros VI

- La función sucesora de aridad uno $S(x) = x + 1$ es calculada por el programa que mueve el contenido del registro 1 (llame a este número [1]) al registro 0 y luego incrementa el número:



- La función de proyección I_n^k es calculada por el programa de siete líneas que mueve el contenido del registro n al registro 0.

Máquinas de registros VII

- A continuación, queremos mostrar que la clase de funciones parciales calculadas por programas de máquinas de registros está cerrada bajo composición.

Máquinas de registros VII

- A continuación, queremos mostrar que la clase de funciones parciales calculadas por programas de máquinas de registros está cerrada bajo composición.
- Es decir, queremos saber que siempre que tengamos programas de máquinas de registros que calculen $f, g_1, \ldots, g_n$ y

$$h(\vec{x}) = f(g_1(\vec{x}), \ldots, g_n(\vec{x})),$$

entonces podemos producir un programa para h.

Máquinas de registros VII

- A continuación, queremos mostrar que la clase de funciones parciales calculadas por programas de máquinas de registros está cerrada bajo composición.
- Es decir, queremos saber que siempre que tengamos programas de máquinas de registros que calculen $f, g_1, \ldots, g_n$ y

$$h(\vec{x}) = f(g_1(\vec{x}), \ldots, g_n(\vec{x})),$$

  entonces podemos producir un programa para h.
- Esto implica encadenar varios programas.

Máquinas de registros VII

- A continuación, queremos mostrar que la clase de funciones parciales calculadas por programas de máquinas de registros está cerrada bajo composición.
- Es decir, queremos saber que siempre que tengamos programas de máquinas de registros que calculen $f, g_1, \ldots, g_n$ y

$$h(\vec{x}) = f(g_1(\vec{x}), \ldots, g_n(\vec{x})),$$

  entonces podemos producir un programa para h.
- Esto implica encadenar varios programas.
- Pero se debe tener cuidado para asegurarse de que un programa no tropiece con la basura dejada por un programa anterior y que no borre los datos necesarios para un programa posterior.

Máquinas de registros VIII

- Sabemos lo que significa que $\mathcal{P}$ calcule f: cuando le proporcionamos a $\mathcal{P}$ las condiciones ideales (la entrada en los registros $1, \ldots, k$, los otros registros están vacíos), entonces $\mathcal{P}$ regresará (en el registro 0) el valor de la función, si está definido.

Máquinas de registros VIII

- Sabemos lo que significa que $\mathcal{P}$ calcule f: cuando le proporcionamos a $\mathcal{P}$ las condiciones ideales (la entrada en los registros $1, \ldots, k$, los otros registros están vacíos), entonces $\mathcal{P}$ regresará (en el registro 0) el valor de la función, si está definido.
- Pero necesitamos un programa que sea menos complicado.

Máquinas de registros VIII

- Sabemos lo que significa que $\mathcal{P}$ calcule f: cuando le proporcionamos a $\mathcal{P}$ las condiciones ideales (la entrada en los registros $1, \ldots, k$, los otros registros están vacíos), entonces $\mathcal{P}$ regresará (en el registro 0) el valor de la función, si está definido.
- Pero necesitamos un programa que sea menos complicado.
- ¿Qué pasa si las condiciones no son ideales?

Máquinas de registros VIII

- Sabemos lo que significa que $\mathcal{P}$ calcule f: cuando le proporcionamos a $\mathcal{P}$ las condiciones ideales (la entrada en los registros $1, \ldots, k$, los otros registros están vacíos), entonces $\mathcal{P}$ regresará (en el registro 0) el valor de la función, si está definido.
- Pero necesitamos un programa que sea menos complicado.
- ¿Qué pasa si las condiciones no son ideales?
- Supongamos que la entrada está en los registros $r_1, \ldots, r_k$, donde estos son k números distintos (no necesariamente consecutivos y no necesariamente en orden creciente). Y supongamos que no queremos prometer que los demás registros estén vacíos.

Máquinas de registros VIII

- Sabemos lo que significa que $\mathcal{P}$ calcule f: cuando le proporcionamos a $\mathcal{P}$ las condiciones ideales (la entrada en los registros $1, \ldots, k$, los otros registros están vacíos), entonces $\mathcal{P}$ regresará (en el registro 0) el valor de la función, si está definido.
- Pero necesitamos un programa que sea menos complicado.
- ¿Qué pasa si las condiciones no son ideales?
- Supongamos que la entrada está en los registros $r_1, \ldots, r_k$, donde estos son k números distintos (no necesariamente consecutivos y no necesariamente en orden creciente). Y supongamos que no queremos prometer que los demás registros estén vacíos.
- Además, queremos un programa que no borra ni altera el contenido de los primeros s registros, para algún número grande s queremos que la información de esos registros se mantenga segura.

Máquinas de registros VIII

- Sabemos lo que significa que $\mathcal{P}$ calcule f: cuando le proporcionamos a $\mathcal{P}$ las condiciones ideales (la entrada en los registros $1, \ldots, k$, los otros registros están vacíos), entonces $\mathcal{P}$ regresará (en el registro 0) el valor de la función, si está definido.
- Pero necesitamos un programa que sea menos complicado.
- ¿Qué pasa si las condiciones no son ideales?
- Supongamos que la entrada está en los registros $r_1, \ldots, r_k$, donde estos son k números distintos (no necesariamente consecutivos y no necesariamente en orden creciente). Y supongamos que no queremos prometer que los demás registros estén vacíos.
- Además, queremos un programa que no borra ni altera el contenido de los primeros s registros, para algún número grande s queremos que la información de esos registros se mantenga segura.
- Esto es lo que queremos, formulado como una definición:

Máquinas de registros IX

- **Definición:** Supongamos que f es una función parcial de aridad k, $\mathcal{Q}$ es un programa, $r_1, \ldots, r_k$ son números naturales distintos, y s y t son números naturales.

Máquinas de registros IX

- **Definición:** Supongamos que f es una función parcial de aridad k, $\mathcal{Q}$ es un programa, $r_1, \ldots, r_k$ son números naturales distintos, y s y t son números naturales.
- Entonces decimos que $\mathcal{Q}$ calcula f a partir de los registros $r_1, \ldots, r_k$ en t preservando s si cada vez que iniciamos una máquina de registros (con suficientes registros) con el programa $\mathcal{Q}$ y con $a_1, \ldots, a_k$ en los registros $r_1, \ldots, r_k$ entonces no importa lo que haya inicialmente en los otros registros, tenemos los siguientes resultados finales:

Máquinas de registros IX

- **Definición:** Supongamos que f es una función parcial de aridad k, $\mathcal{Q}$ es un programa, $r_1, \ldots, r_k$ son números naturales distintos, y s y t son números naturales.
- Entonces decimos que $\mathcal{Q}$ calcula f a partir de los registros $r_1, \ldots, r_k$ en t preservando s si cada vez que iniciamos una máquina de registros (con suficientes registros) con el programa $\mathcal{Q}$ y con $a_1, \ldots, a_k$ en los registros $r_1, \ldots, r_k$ entonces no importa lo que haya inicialmente en los otros registros, tenemos los siguientes resultados finales:
  - Si $f(a_1, \ldots, a_k)$ está definida, entonces el cálculo finalmente se detiene (es decir, se detiene buscando la primera instrucción inexistente, la $(q+1)$-ésima instrucción, donde q es la longitud de $\mathcal{Q}$), con el valor $f(a_1, \ldots, a_k)$ en t. Además, los primeros s registros, los registros $0, 1, \ldots, s-1$, contienen los mismos números que tenían inicialmente, excepto posiblemente el de t.

Máquinas de registros IX

- **Definición:** Supongamos que f es una función parcial de aridad k, $\mathcal{Q}$ es un programa, $r_1, \ldots, r_k$ son números naturales distintos, y s y t son números naturales.
- Entonces decimos que $\mathcal{Q}$ calcula f a partir de los registros $r_1, \ldots, r_k$ en t preservando s si cada vez que iniciamos una máquina de registros (con suficientes registros) con el programa $\mathcal{Q}$ y con $a_1, \ldots, a_k$ en los registros $r_1, \ldots, r_k$ entonces no importa lo que haya inicialmente en los otros registros, tenemos los siguientes resultados finales:
  - Si $f(a_1, \ldots, a_k)$ está definida, entonces el cálculo finalmente se detiene (es decir, se detiene buscando la primera instrucción inexistente, la $(q+1)$-ésima instrucción, donde q es la longitud de $\mathcal{Q}$), con el valor $f(a_1, \ldots, a_k)$ en t. Además, los primeros s registros, los registros $0, 1, \ldots, s-1$, contienen los mismos números que tenían inicialmente, excepto posiblemente el de t.
  - Si $f(a_1, \ldots, a_k)$ no está definida, entonces el cálculo nunca se detiene.

Máquinas de registros X

- Como caso especial, podemos decir que si $\mathcal{Q}$ calcula f a partir de $1, \ldots, k$ en 0 preservando 0, entonces $\mathcal{Q}$ calcula f (en el sentido definido originalmente).

Máquinas de registros X

- Como caso especial, podemos decir que si $\mathcal{Q}$ calcula f a partir de $1, \ldots, k$ en 0 preservando 0, entonces $\mathcal{Q}$ calcula f (en el sentido definido originalmente).
- Lo contrario no es del todo cierto debido a la cuestión de si los registros están inicialmente vacíos.

Máquinas de registros X

- Como caso especial, podemos decir que si $\mathcal{Q}$ calcula f a partir de $1, \ldots, k$ en 0 preservando 0, entonces $\mathcal{Q}$ calcula f (en el sentido definido originalmente).
- Lo contrario no es del todo cierto debido a la cuestión de si los registros están inicialmente vacíos.
- El siguiente lema dice que podemos tener lo que pide la definición anterior.

Máquinas de registros X

- Como caso especial, podemos decir que si $\mathcal{Q}$ calcula f a partir de $1, \ldots, k$ en 0 preservando 0, entonces $\mathcal{Q}$ calcula f (en el sentido definido originalmente).
- Lo contrario no es del todo cierto debido a la cuestión de si los registros están inicialmente vacíos.
- El siguiente lema dice que podemos tener lo que pide la definición anterior.
- **Lema:** Supongamos que el programa $\mathcal{P}$ calcula la función parcial f de aridad k. Sean $r_1, \ldots, r_k$ números naturales distintos. Sean t y s números naturales. Entonces podemos encontrar un programa $\mathcal{Q}$ que calcule f a partir de $r_1, \ldots, r_k$ en t preservando s.

Máquinas de registros XI

- *Prueba:* Sea M la *dirección* más grande en $\mathcal{P}$ (es decir, el número más grande tal que alguna instrucción de incremento o decremento en $\mathcal{P}$ direcciona el registro M).

Máquinas de registros XI

- *Prueba:* Sea M la *dirección* más grande en $\mathcal{P}$ (es decir, el número más grande tal que alguna instrucción de incremento o decremento en $\mathcal{P}$ direcciona el registro M).
- Probablemente $M > k$; si no luego aumente M para que sea $k + 1$. Sea j el mayor de los números $s, r_1, \ldots, r_k$.

Máquinas de registros XI

- *Prueba:* Sea M la *dirección* más grande en $\mathcal{P}$ (es decir, el número más grande tal que alguna instrucción de incremento o decremento en $\mathcal{P}$ direcciona el registro M).
- Probablemente $M > k$; si no luego aumente M para que sea $k + 1$. Sea j el mayor de los números $s, r_1, \ldots, r_k$.
- Aquí está el programa $\mathcal{Q}$:

COPY	r_1 to $j + 1$ usando $j + 2$
COPY	r_2 to $j + 2$ usando $j + 3$
$\ldots$	
COPY	r_k to $j + k$ usando $j + k + 1$
CLEAR	j
CLEAR	$j + k + 1$
CLEAR	$j + k + 2$
$\ldots$	
CLEAR	$j + M$
$\mathcal{P}$	reubicado por j
MOVE	de j a t (si $j \neq t$)

Máquinas de registros XII

- Aquí, "reubicado por j" significa que j se agrega a la dirección de todas las instrucciones de incremento y decremento.

Máquinas de registros XII

- Aquí, "reubicado por j" significa que j se agrega a la dirección de todas las instrucciones de incremento y decremento.
- Por tanto, el programa reubicado opera en los registros $j, j+1, \ldots, j+M$ exactamente como $\mathcal{P}$ operó en los registros $0, 1, \ldots, M$.

Máquinas de registros XII

- Aquí, "reubicado por j" significa que j se agrega a la dirección de todas las instrucciones de incremento y decremento.
- Por tanto, el programa reubicado opera en los registros $j, j+1, \ldots, j+M$ exactamente como $\mathcal{P}$ operó en los registros $0, 1, \ldots, M$.
- Dado que $\mathcal{P}$ calcula f, el programa reubicado dejará $f(a_1, \ldots, a_k)$, si está definido, en el registro j.

Máquinas de registros XII

- Aquí, "reubicado por j" significa que j se agrega a la dirección de todas las instrucciones de incremento y decremento.
- Por tanto, el programa reubicado opera en los registros $j, j+1, \ldots, j+M$ exactamente como $\mathcal{P}$ operó en los registros $0, 1, \ldots, M$.
- Dado que $\mathcal{P}$ calcula f, el programa reubicado dejará $f(a_1, \ldots, a_k)$, si está definido, en el registro j.
- El programa $\mathcal{Q}$ entonces mueve j al registro t.

Máquinas de registros XII

- Aquí, "reubicado por j" significa que j se agrega a la dirección de todas las instrucciones de incremento y decremento.
- Por tanto, el programa reubicado opera en los registros $j, j+1, \ldots, j+M$ exactamente como $\mathcal{P}$ operó en los registros $0, 1, \ldots, M$.
- Dado que $\mathcal{P}$ calcula f, el programa reubicado dejará $f(a_1, \ldots, a_k)$, si está definido, en el registro j.
- El programa $\mathcal{Q}$ entonces mueve j al registro t.
- Exceptuando al registro t, el programa deja los registros $0, 1, \ldots, j-1$ sin alterar. El siguiente mapa ilustra cómo $\mathcal{Q}$ usa los registros:

registro	0	intacto
registro	1	intacto
	$\vdots$	
registro	$j-1$	intacto
registro	j	salida
registro	$j+1$	entrada
	$\vdots$	
registro	$j+k$	entrada
registro	$j+k+1$	espacio de trabajo
	$\vdots$	
registro	$j+M$	espacio de trabajo

Máquinas de registros XIII

- El lema será una herramienta útil siempre que necesitemos unir diferentes programas.

Máquinas de registros XIII

- El lema será una herramienta útil siempre que necesitemos unir diferentes programas.
- Como nuestra primera aplicación de este lema, podemos demostrar que la clase de funciones parciales computables por máquinas de registros es cerrada bajo composición.

Máquinas de registros XIII

- El lema será una herramienta útil siempre que necesitemos unir diferentes programas.
- Como nuestra primera aplicación de este lema, podemos demostrar que la clase de funciones parciales computables por máquinas de registros es cerrada bajo composición.
- **Teorema:** La clase de funciones parciales computables por máquinas de registros es cerrada bajo composición. Es decir, siempre que tengamos programas de máquinas de registros que calculen funciones parciales $f, g_1, \ldots, g_n$ de las cuales se obtiene h por composición, entonces podemos hacer un programa que calcula la función parcial h.

Máquinas de registros XIV

- *Prueba:* Supongamos que la función parcial h de aridad k se obtiene por composición de f y $g_1, \ldots, g_n$:

$$h(\vec{x}) = f(g_1(\vec{x}), \ldots, g_n(\vec{x})).$$

Máquinas de registros XIV

- *Prueba:* Supongamos que la función parcial h de aridad k se obtiene por composición de f y $g_1, \ldots, g_n$:

$$h(\vec{x}) = f(g_1(\vec{x}), \ldots, g_n(\vec{x})).$$

- Supongamos además que tenemos programas que calculan $f, g_1, \ldots, g_n$. queremos hacer un programa que calcule h. El cual es el siguiente:

Calcule g_1 a partir de $1, \ldots, k$ a $k+1$, conservando $k+1$.

Calcule g_2 a partir de $1, \ldots, k$ a $k+2$, conservando $k+2$.

$\ldots$

Calcule g_n a partir de $1, \ldots, k$ a $k+n$, conservando $k+n$.

Calcule f a partir de $k+1, \ldots, k+n$ a 0, conservando 0.

Máquinas de registros XIV

- *Prueba:* Supongamos que la función parcial h de aridad k se obtiene por composición de f y $g_1, \ldots, g_n$:

$$h(\vec{x}) = f(g_1(\vec{x}), \ldots, g_n(\vec{x})).$$

- Supongamos además que tenemos programas que calculan $f, g_1, \ldots, g_n$. queremos hacer un programa que calcule h. El cual es el siguiente:

$$\begin{aligned}
&\text{Calcule } g_1 \text{ a partir de } 1, \ldots, k \text{ a } k+1, \text{ conservando } k+1.\\
&\text{Calcule } g_2 \text{ a partir de } 1, \ldots, k \text{ a } k+2, \text{ conservando } k+2.\\
&\qquad \ldots\\
&\text{Calcule } g_n \text{ a partir de } 1, \ldots, k \text{ a } k+n, \text{ conservando } k+n.\\
&\text{Calcule } f \text{ a partir de } k+1, \ldots, k+n \text{ a } 0, \text{ conservando } 0.
\end{aligned}$$

- Aquí, nos basamos en el lema para proporcionar los componentes del programa.

Máquinas de registros XIV

- *Prueba:* Supongamos que la función parcial h de aridad k se obtiene por composición de f y $g_1, \ldots, g_n$:

$$h(\vec{x}) = f(g_1(\vec{x}), \ldots, g_n(\vec{x})).$$

- Supongamos además que tenemos programas que calculan $f, g_1, \ldots, g_n$. queremos hacer un programa que calcule h. El cual es el siguiente:

$$\begin{aligned}
&\text{Calcule } g_1 \text{ a partir de } 1, \ldots, k \text{ a } k+1, \text{ conservando } k+1.\\
&\text{Calcule } g_2 \text{ a partir de } 1, \ldots, k \text{ a } k+2, \text{ conservando } k+2.\\
&\qquad \ldots\\
&\text{Calcule } g_n \text{ a partir de } 1, \ldots, k \text{ a } k+n, \text{ conservando } k+n.\\
&\text{Calcule } f \text{ a partir de } k+1, \ldots, k+n \text{ a } 0, \text{ conservando } 0.
\end{aligned}$$

- Aquí, nos basamos en el lema para proporcionar los componentes del programa.
- Observe que para que el programa se detenga, necesitamos que las funciones $g_1(\vec{x}), \ldots, g_n(\vec{x})$ estén definidas y necesitamos que f esté definida en $g_1(\vec{x}), \ldots, g_n(\vec{x})$.

$\dashv$

Máquinas de registros XV

- Por ejemplo, supongamos que h está dada por la ecuación $h(x, y, z) = f(g(z, y), 7, x)$ y tenemos programas de registros para f y g.

Máquinas de registros XV

- Por ejemplo, supongamos que h está dada por la ecuación $h(x, y, z) = f(g(z, y), 7, x)$ y tenemos programas de registros para f y g.
- Del teorema anterior se deduce que h es computable por alguna máquina de registros. Podemos escribir

$$h(x, y, z) = f(g(I_3^3(x, y, z), I_2^3(x, y, z)), K_7(x, y, z), I_1^3(x, y, z))$$

(donde K_7 es una función constante) y aplicar el teorema dos veces, primero para obtener $g(I_3^3(x, y, z), I_2^3(x, y, z))$ y luego para obtener h.

Máquinas de registros XV

- Por ejemplo, supongamos que h está dada por la ecuación $h(x, y, z) = f(g(z, y), 7, x)$ y tenemos programas de registros para f y g.
- Del teorema anterior se deduce que h es computable por alguna máquina de registros. Podemos escribir

$$h(x, y, z) = f(g(I_3^3(x, y, z), I_2^3(x, y, z)), K_7(x, y, z), I_1^3(x, y, z))$$

  (donde K_7 es una función constante) y aplicar el teorema dos veces, primero para obtener $g(I_3^3(x, y, z), I_2^3(x, y, z))$ y luego para obtener h.
- La moraleja de este ejemplo es que podemos poner libremente las variables donde queramos y aplicar composición con funciones de proyección para justificar lo que hemos hecho.

Máquinas de registros XV

- Por ejemplo, supongamos que h está dada por la ecuación $h(x, y, z) = f(g(z, y), 7, x)$ y tenemos programas de registros para f y g.
- Del teorema anterior se deduce que h es computable por alguna máquina de registros. Podemos escribir

$$h(x, y, z) = f(g(I_3^3(x, y, z), I_2^3(x, y, z)), K_7(x, y, z), I_1^3(x, y, z))$$

  (donde K_7 es una función constante) y aplicar el teorema dos veces, primero para obtener $g(I_3^3(x, y, z), I_2^3(x, y, z))$ y luego para obtener h.
- La moraleja de este ejemplo es que podemos poner libremente las variables donde queramos y aplicar composición con funciones de proyección para justificar lo que hemos hecho.
- En particular, si

$$h(x_1, x_2, \ldots, x_m) = f(\_\_\_, \_\_\_, \ldots, \_\_\_),$$

  donde cada espacio en blanco se llena con algún x_i o alguna constante, entonces a partir de un programa para f podemos obtener un programa para h.

Máquinas de registros XVI

- Aún así, este capítulo aún no ha producido un programa para una sola función “interesante”.

Máquinas de registros XVI

- Aún así, este capítulo aún no ha producido un programa para una sola función “interesante”.
- Para eso, necesitamos un resultado de cerradura más: cerradura bajo recursividad primitiva.

Máquinas de registros XVI

- Aún así, este capítulo aún no ha producido un programa para una sola función "interesante".
- Para eso, necesitamos un resultado de cerradura más: cerradura bajo recursividad primitiva.
- Es decir, queremos saber que si h se obtiene de f y g mediante recursividad primitiva

$$h(\vec{x}, 0) = f(\vec{x})$$
$$h(\vec{x}, y+1) = g(h(\vec{x}, y), \vec{x}, y)$$

y tenemos programas de registros para f y g, entonces podemos obtener un programa para h.

Máquinas de registros XVI

- Aún así, este capítulo aún no ha producido un programa para una sola función "interesante".
- Para eso, necesitamos un resultado de cerradura más: cerradura bajo recursividad primitiva.
- Es decir, queremos saber que si h se obtiene de f y g mediante recursividad primitiva

$$h(\vec{x}, 0) = f(\vec{x})$$
$$h(\vec{x}, y+1) = g(h(\vec{x}, y), \vec{x}, y)$$

  y tenemos programas de registros para f y g, entonces podemos obtener un programa para h.
- O en el caso de que $\vec{x}$ esté vacío

$$h(0) = m$$
$$h(y+1) = g(h(y), y)$$

  (para algún número m) y tenemos un programa para g, entonces queremos saber si podemos obtener un programa para h.

Máquinas de registros XVII

- De ello se deducirá que todas las funciones recursivas primitivas (en particular, las del Capítulo 2) son computables por máquinas de registros.

Máquinas de registros XVII

- De ello se deducirá que todas las funciones recursivas primitivas (en particular, las del Capítulo 2) son computables por máquinas de registros.
- Y finalmente, veremos que la clase de funciones computables de máquinas de registros incluye mucho más que los ejemplos simplistas con los que comenzamos.

Máquinas de registros XVII

- De ello se deducirá que todas las funciones recursivas primitivas (en particular, las del Capítulo 2) son computables por máquinas de registros.
- Y finalmente, veremos que la clase de funciones computables de máquinas de registros incluye mucho más que los ejemplos simplistas con los que comenzamos.
- **Teorema:** La clase de funciones parciales computables por máquinas de registros es cerrada bajo recursividad primitiva.

Máquinas de registros XVII

- De ello se deducirá que todas las funciones recursivas primitivas (en particular, las del Capítulo 2) son computables por máquinas de registros.
- Y finalmente, veremos que la clase de funciones computables de máquinas de registros incluye mucho más que los ejemplos simplistas con los que comenzamos.
- **Teorema:** La clase de funciones parciales computables por máquinas de registros es cerrada bajo recursividad primitiva.
- *Prueba:* Supongamos que h es la función parcial de aridad $(n+1)$ obtenida por recursión primitiva a partir de las funciones parciales f y g:

$$h(\vec{x}, 0) = f(\vec{x})$$
$$h(\vec{x}, y+1) = g(h(\vec{x}, y), \vec{x}, y)$$

Máquinas de registros XVII

- De ello se deducirá que todas las funciones recursivas primitivas (en particular, las del Capítulo 2) son computables por máquinas de registros.
- Y finalmente, veremos que la clase de funciones computables de máquinas de registros incluye mucho más que los ejemplos simplistas con los que comenzamos.
- **Teorema:** La clase de funciones parciales computables por máquinas de registros es cerrada bajo recursividad primitiva.
- *Prueba:* Supongamos que h es la función parcial de aridad $(n+1)$ obtenida por recursión primitiva a partir de las funciones parciales f y g:

$$h(\vec{x}, 0) = f(\vec{x})$$
$$h(\vec{x}, y+1) = g(h(\vec{x}, y), \vec{x}, y)$$

- Supongamos que tenemos programas de registros para f y g.

Máquinas de registros XVII

- De ello se deducirá que todas las funciones recursivas primitivas (en particular, las del Capítulo 2) son computables por máquinas de registros.
- Y finalmente, veremos que la clase de funciones computables de máquinas de registros incluye mucho más que los ejemplos simplistas con los que comenzamos.
- **Teorema:** La clase de funciones parciales computables por máquinas de registros es cerrada bajo recursividad primitiva.
- *Prueba:* Supongamos que h es la función parcial de aridad $(n+1)$ obtenida por recursión primitiva a partir de las funciones parciales f y g:

$$h(\vec{x}, 0) = f(\vec{x})$$
$$h(\vec{x}, y+1) = g(h(\vec{x}, y), \vec{x}, y)$$

- Supongamos que tenemos programas de registros para f y g.
- Necesitamos hacer un programa para h.

Máquinas de registros XVIII

- El programa comenzará con $x_1, \ldots, x_n, y$ en los registros $1, 2, \ldots, n, n+1$.

Máquinas de registros XVIII

- El programa comenzará con $x_1, \ldots, x_n, y$ en los registros $1, 2, \ldots, n, n+1$.
- El programa pondrá $h(\vec{x}, t)$ en el registro 0 primero para $t = 0$, luego para $t = 1$, y así sucesivamente hasta $t = y$.

Máquinas de registros XVIII

- El programa comenzará con $x_1, \ldots, x_n, y$ en los registros $1, 2, \ldots, n, n+1$.
- El programa pondrá $h(\vec{x}, t)$ en el registro 0 primero para $t = 0$, luego para $t = 1$, y así sucesivamente hasta $t = y$.
- El número t se mantiene en el registro $n + 2$, que inicialmente contiene 0.

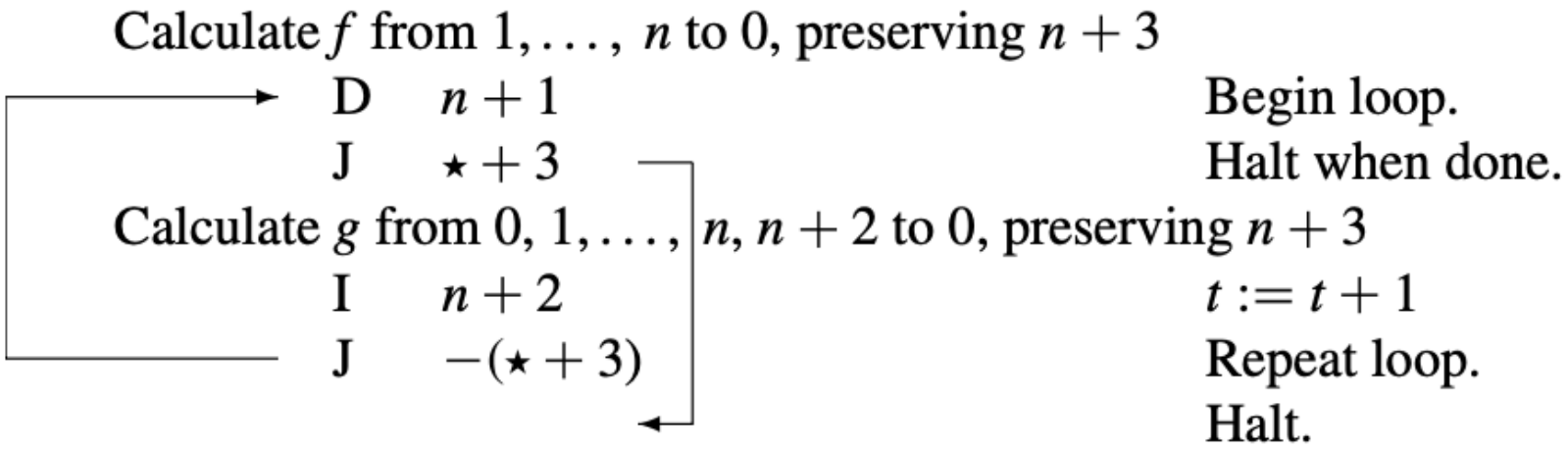
Máquinas de registros XVIII

- El programa comenzará con $x_1, \ldots, x_n, y$ en los registros $1, 2, \ldots, n, n+1$.
- El programa pondrá $h(\vec{x}, t)$ en el registro 0 primero para $t = 0$, luego para $t = 1$, y así sucesivamente hasta $t = y$.
- El número t se mantiene en el registro $n + 2$, que inicialmente contiene 0.
- El siguiente mapa ilustra este uso:

registro	0	$h(\vec{x}, t)$
registro	1	x_1
	$\vdots$	
registro	n	x_n
registro	$n+1$	$y - t$
registro	$n+2$	t
registro	$n+3$	espacio de trabajo
	$\vdots$	

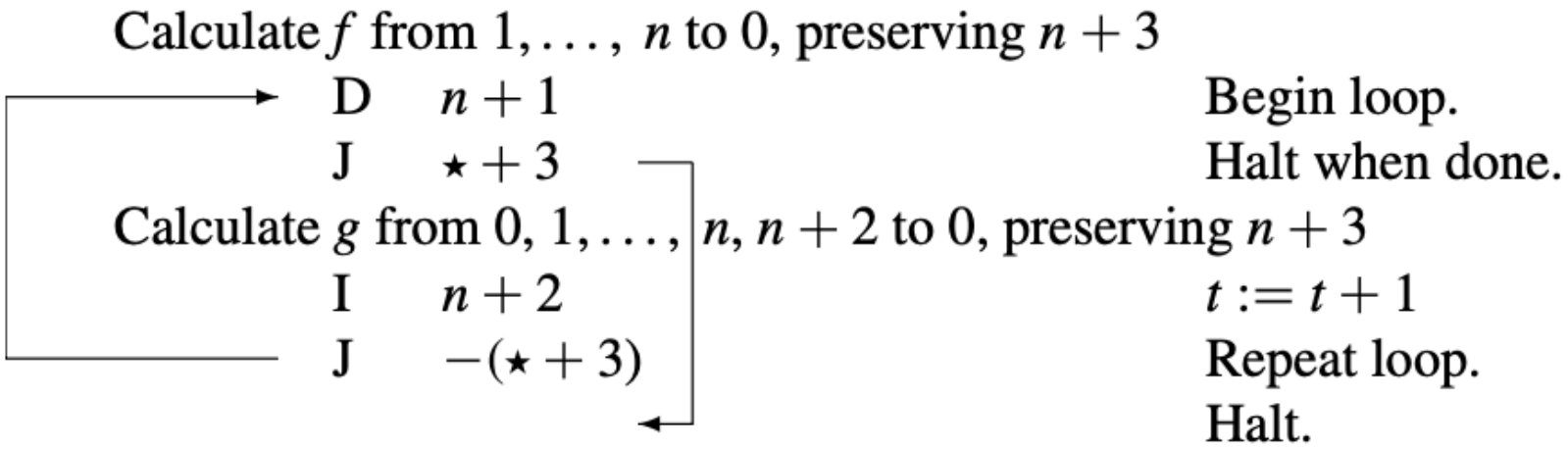
Máquinas de registros XIX

- Aquí está el programa:



Máquinas de registros XIX

- Aquí está el programa:



- Aquí $\star$ es la longitud del programa que se utiliza para g.

Máquinas de registros XX

- Para ver la corrección de este programa, establecemos lo siguiente:

Máquinas de registros XX

- Para ver la corrección de este programa, establecemos lo siguiente:
- Afirmación: Cada vez que llegamos a la instrucción D $n+1$ (al comienzo del ciclo), después de ejecutar el ciclo k veces,

Máquinas de registros XX

- Para ver la corrección de este programa, establecemos lo siguiente:
- Afirmación: Cada vez que llegamos a la instrucción D $n+1$ (al comienzo del ciclo), después de ejecutar el ciclo k veces,
  - el registro 0 contiene $h(\vec{x}, k)$,

Máquinas de registros XX

- Para ver la corrección de este programa, establecemos lo siguiente:
- Afirmación: Cada vez que llegamos a la instrucción D $n+1$ (al comienzo del ciclo), después de ejecutar el ciclo k veces,
  - el registro 0 contiene $h(\vec{x}, k)$,
  - el registro $n+1$ contiene $y-k$,

Máquinas de registros XX

- Para ver la corrección de este programa, establecemos lo siguiente:
- Afirmación: Cada vez que llegamos a la instrucción D $n+1$ (al comienzo del ciclo), después de ejecutar el ciclo k veces,
  - el registro 0 contiene $h(\vec{x}, k)$,
  - el registro $n+1$ contiene $y-k$,
  - el registro $n+2$ contiene k.

Máquinas de registros XX

- Para ver la corrección de este programa, establecemos lo siguiente:
- Afirmación: Cada vez que llegamos a la instrucción D $n+1$ (al comienzo del ciclo), después de ejecutar el ciclo k veces,
  - el registro 0 contiene $h(\vec{x}, k)$,
  - el registro $n+1$ contiene $y-k$,
  - el registro $n+2$ contiene k.
- La afirmación de que existen "invariantes de ciclo" se demuestra por inducción (como es habitual en programas con ciclos) sobre k.

Máquinas de registros XXI

- Para $k = 0$, cuando lleguemos a D $n + 1$ sin haber ejecutado el bucle en absoluto, el registro 0 contiene $f(\vec{x})$, que es $h(x, 0)$, el registro $n + 1$ no se modifica, por lo que todavía contiene y, y el registro $n + 2$ sigue siendo 0.

Máquinas de registros XXI

- Para $k = 0$, cuando lleguemos a D $n + 1$ sin haber ejecutado el bucle en absoluto, el registro 0 contiene $f(\vec{x})$, que es $h(x, 0)$, el registro $n + 1$ no se modifica, por lo que todavía contiene y, y el registro $n + 2$ sigue siendo 0.
- Ahora vamos al paso inductivo. En el $(k + 1)$-ésimo paso por el ciclo, decrementamos el registro $n + 1$ (según la hipótesis inductiva, anteriormente contenía $y - k$, por lo que ahora es $y - (k + 1)$), incrementamos el registro $n + 2$ (anteriormente contenía k, por lo que ahora es $(k + 1)$, y en el registro 0 ponemos $g(h(\vec{x}, k), \vec{x}, k)$, que de hecho es $h(\vec{x}, k + 1)$.

Máquinas de registros XXI

- Para $k = 0$, cuando lleguemos a D $n + 1$ sin haber ejecutado el bucle en absoluto, el registro 0 contiene $f(\vec{x})$, que es $h(x, 0)$, el registro $n + 1$ no se modifica, por lo que todavía contiene y, y el registro $n + 2$ sigue siendo 0.
- Ahora vamos al paso inductivo. En el $(k + 1)$-ésimo paso por el ciclo, decrementamos el registro $n + 1$ (según la hipótesis inductiva, anteriormente contenía $y - k$, por lo que ahora es $y - (k + 1)$), incrementamos el registro $n + 2$ (anteriormente contenía k, por lo que ahora es $(k + 1)$, y en el registro 0 ponemos $g(h(\vec{x}, k), \vec{x}, k)$, que de hecho es $h(\vec{x}, k + 1)$.
- Entonces, por inducción, la afirmación se cumple cada vez que iniciamos el ciclo. El programa se detiene cuando iniciamos el ciclo con 0 en el registro $n + 1$.

Máquinas de registros XXI

- Para $k = 0$, cuando lleguemos a D $n + 1$ sin haber ejecutado el bucle en absoluto, el registro 0 contiene $f(\vec{x})$, que es $h(x, 0)$, el registro $n + 1$ no se modifica, por lo que todavía contiene y, y el registro $n + 2$ sigue siendo 0.
- Ahora vamos al paso inductivo. En el $(k + 1)$-ésimo paso por el ciclo, decrementamos el registro $n + 1$ (según la hipótesis inductiva, anteriormente contenía $y - k$, por lo que ahora es $y - (k + 1)$), incrementamos el registro $n + 2$ (anteriormente contenía k, por lo que ahora es $(k + 1)$, y en el registro 0 ponemos $g(h(\vec{x}, k), \vec{x}, k)$, que de hecho es $h(\vec{x}, k + 1)$.
- Entonces, por inducción, la afirmación se cumple cada vez que iniciamos el ciclo. El programa se detiene cuando iniciamos el ciclo con 0 en el registro $n + 1$.
- En este punto, hemos hecho el ciclo y veces (por la afirmación), y el registro 0 contiene $h(\vec{x}, y)$, como se deseaba.

Máquinas de registros XXI

- Para $k = 0$, cuando lleguemos a D $n + 1$ sin haber ejecutado el bucle en absoluto, el registro 0 contiene $f(\vec{x})$, que es $h(x, 0)$, el registro $n + 1$ no se modifica, por lo que todavía contiene y, y el registro $n + 2$ sigue siendo 0.
- Ahora vamos al paso inductivo. En el $(k + 1)$-ésimo paso por el ciclo, decrementamos el registro $n + 1$ (según la hipótesis inductiva, anteriormente contenía $y - k$, por lo que ahora es $y - (k + 1)$), incrementamos el registro $n + 2$ (anteriormente contenía k, por lo que ahora es $(k + 1)$, y en el registro 0 ponemos $g(h(\vec{x}, k), \vec{x}, k)$, que de hecho es $h(\vec{x}, k + 1)$.
- Entonces, por inducción, la afirmación se cumple cada vez que iniciamos el ciclo. El programa se detiene cuando iniciamos el ciclo con 0 en el registro $n + 1$.
- En este punto, hemos hecho el ciclo y veces (por la afirmación), y el registro 0 contiene $h(\vec{x}, y)$, como se deseaba.
- El programa se modifica fácilmente para el caso en que $\vec{x}$ sea vacío.

Máquinas de registros XXII

- Queremos usar los registros de la siguiente manera:

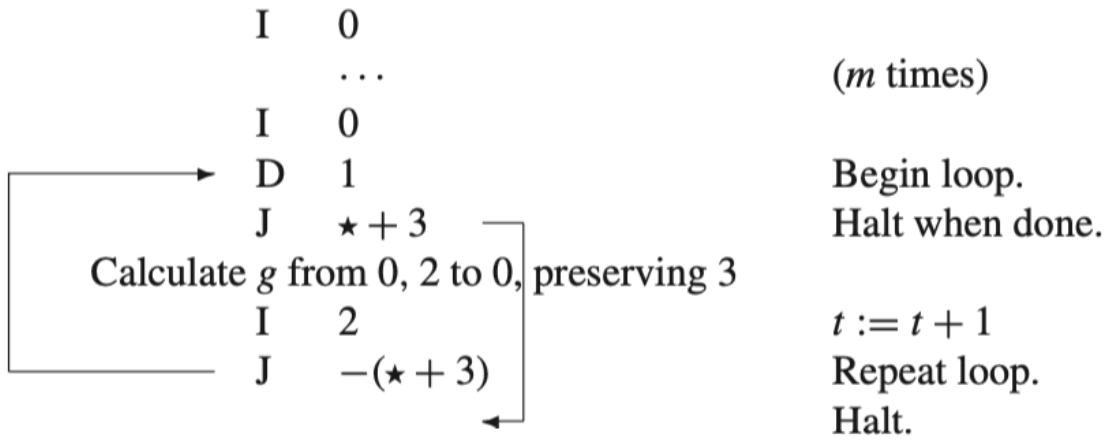
registro	0	$h(t)$
registro	1	$y - t$
registro	2	t
registro	3	espacio de trabajo
	$\vdots$	

Máquinas de registros XXII

- Queremos usar los registros de la siguiente manera:

registro	0	$h(t)$
registro	1	$y - t$
registro	2	t
registro	3	espacio de trabajo
	$\vdots$	

- Aquí está el programa:

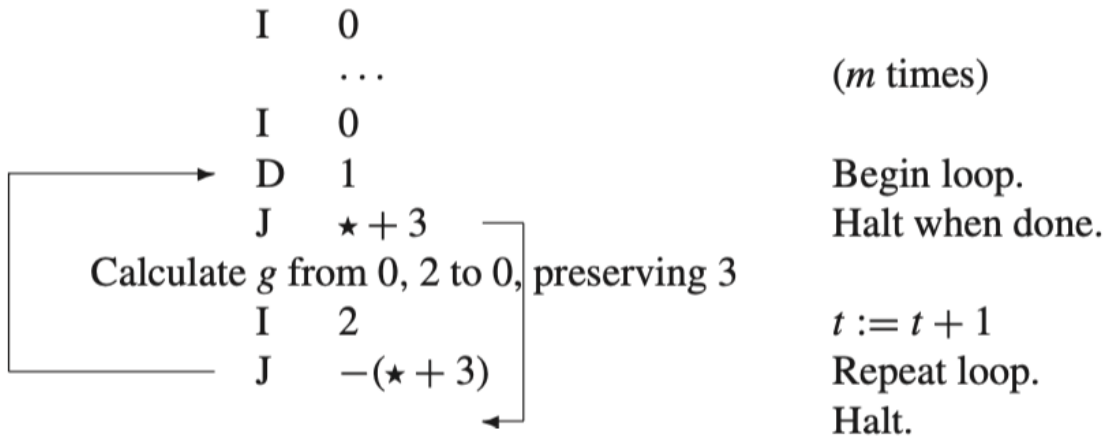


Máquinas de registros XXII

- Queremos usar los registros de la siguiente manera:

registro	0	$h(t)$
registro	1	$y - t$
registro	2	t
registro	3	espacio de trabajo
	$\vdots$	

- Aquí está el programa:



- El argumento de la corrección sigue siendo aplicable con $n = 0$. ⊣

Máquinas de registros XXIII

- Reuniendo los resultados hasta el momento, llegamos a la siguiente conclusión.

  Teorema: Toda función recursiva primitiva es computable por máquinas de registros.

Máquinas de registros XXIII

- Reuniendo los resultados hasta el momento, llegamos a la siguiente conclusión.

  Teorema: Toda función recursiva primitiva es computable por máquinas de registros.

- Este teorema promete programas de máquinas de registros para todas las funciones recursivas primitivas del capítulo anterior.

Máquinas de registros XXIII

- Reuniendo los resultados hasta el momento, llegamos a la siguiente conclusión.

  Teorema: Toda función recursiva primitiva es computable por máquinas de registros.

- Este teorema promete programas de máquinas de registros para todas las funciones recursivas primitivas del capítulo anterior.

- Pero, por supuesto, podemos hacerlo mejor:

  Teorema: Toda función parcial recursiva general es computable por máquinas de registros.

Máquinas de registros XXIII

- Reuniendo los resultados hasta el momento, llegamos a la siguiente conclusión.

  Teorema: Toda función recursiva primitiva es computable por máquinas de registros.

- Este teorema promete programas de máquinas de registros para todas las funciones recursivas primitivas del capítulo anterior.

- Pero, por supuesto, podemos hacerlo mejor:

  Teorema: Toda función parcial recursiva general es computable por máquinas de registros.

- *Prueba:* Necesitamos agregar el operador-μ

$$h(\vec{x}) = \mu y[g(\vec{x}, y) = 0].$$

Máquinas de registros XXIII

- Reuniendo los resultados hasta el momento, llegamos a la siguiente conclusión.

  Teorema: Toda función recursiva primitiva es computable por máquinas de registros.

- Este teorema promete programas de máquinas de registros para todas las funciones recursivas primitivas del capítulo anterior.

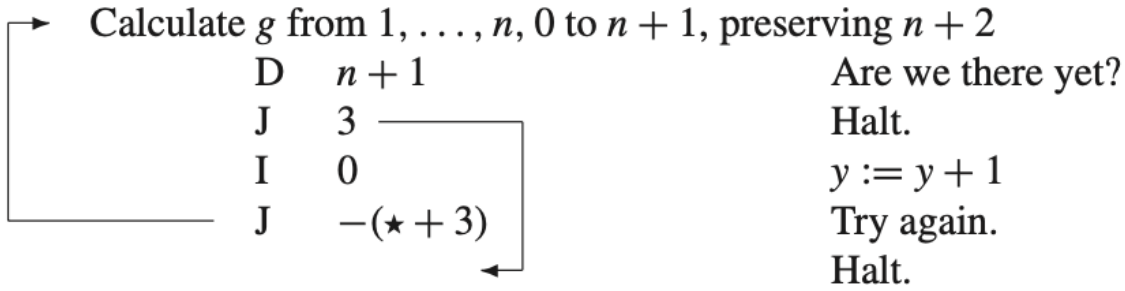
- Pero, por supuesto, podemos hacerlo mejor:

  Teorema: Toda función parcial recursiva general es computable por máquinas de registros.

- *Prueba:* Necesitamos agregar el operador-μ

$$h(\vec{x}) = \mu y[g(\vec{x}, y) = 0].$$

- Usamos el programa obvio:



Máquinas de registros XXIV

- El programa utiliza los registros de la siguiente manera:

registro	0	y
registro	1	x_1
	$\vdots$	
registro	n	x_n
registro	$n+1$	$g(\vec{x}, y)$
registro	$n+2$	espacio de trabajo
	$\vdots$	

Máquinas de registros XXIV

- El programa utiliza los registros de la siguiente manera:

registro	0	y
registro	1	x_1
	$\vdots$	
registro	n	x_n
registro	$n+1$	$g(\vec{x}, y)$
registro	$n+2$	espacio de trabajo
	$\vdots$	

- Por supuesto, es posible que este programa nunca se detenga. $\dashv$

Máquinas de registros XXV

- Este teorema da la mitad de un hecho significativo, formalizar el concepto de calculabilidad efectiva mediante recursividad general y formalizar el concepto de calculabilidad efectiva mediante máquinas de registros conduce a exactamente la misma clase de funciones parciales.

Máquinas de registros XXV

- Este teorema da la mitad de un hecho significativo, formalizar el concepto de calculabilidad efectiva mediante recursividad general y formalizar el concepto de calculabilidad efectiva mediante máquinas de registros conduce a exactamente la misma clase de funciones parciales.
- Pronto llegaremos a la otra mitad.

Máquinas de registros XXV

- Este teorema da la mitad de un hecho significativo, formalizar el concepto de calculabilidad efectiva mediante recursividad general y formalizar el concepto de calculabilidad efectiva mediante máquinas de registros conduce a exactamente la misma clase de funciones parciales.
- Pronto llegaremos a la otra mitad.
- En particular, el teorema ilustra que las máquinas de registros son capaces de hacer mucho más de lo que podría parecer, inicialmente, a partir de su muy simple definición.

Ejercicios

De un programa de máquina de registros que calcule cada una de las siguientes funciones.

1. $$Z(x) = \begin{cases} 1 & \text{si } x = 0 \\ 0 & \text{si } x > 0 \end{cases}$$
2. $$prod(x, y) = x \cdot y$$
3. $$exp(x, y) = x^y$$