Escuela Técnica Superior de Ingeniería **Informática**

etsinf

DSIIC
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Computabilidad y Complejidad

Tema 4: Máquinas de Registros.

Índice:

Bibliografía Básica Recomendada

- Computation. Finite and Infinite Machines (M.L. Minsky)

Generalidades

En este tema describiremos diferentes modelos de cálculo basados estructuralmente en los computadores digitales ordinarios.

Estos modelos consisten en máquinas de registros que abstraen e idealizan los elementos constituyentes de los computadores digitales, de modo que, por su idealización, no tienen existencia real y permiten, al igual que la máquina de Turing, definir la familia de las funciones computables. De hecho todos los modelos que describiremos son computacionalmente equivalentes al modelo de Turing.

Los modelos descritos a continuación son:

- el modelo `RAM`,
- el modelo `RASP`,
- el modelo de la `máquina contador`.

El modelo RAM

La RAM (Random Access Machine) se compone de:

- Una memoria consistente en un número ilimitado de registros. Cada uno de ellos tiene asociada una dirección consistente en un número natural mediante el cual se puede directamente acceder a su contenido: un número natural. El conjunto de direcciones coincide con el conjunto de los números naturales.

- Una unidad de control donde se encuentra el programa a ejecutar junto con el contador de programa. El programa se compone de una secuencia finita de instrucciones pertenecientes a un conjunto finito de instrucciones muy básicas y sencillas. Este modelo de máquina tiene instrucciones con direccionamiento indirecto. Las instrucciones se encuentran numeradas consecutivamente empezando en 0. La ejecución siempre comienza en la primera instrucción y si termina lo hace cuando se intenta acceder a una instrucción inexistente.

La arquitectura de una RAM se corresponde con la arquitectura de Harvard.

Cada registro será designado mediante la notación: R_i, $i \in N$; donde i es su dirección. Su contenido lo denotaremos mediante: $[i]$.

Esta máquina la utilizaremos para computar funciones numéricas de la forma

$$g: N^m \longrightarrow N^k$$

Para su cómputo la máquina arranca con los m primeros registros con los argumentos de la función y el resto de registros con valor nulo. Cuando la máquina termine el cómputo, su resultado será el contenido de los k primeros registros; en otro caso, si el cómputo no termina el resultado no está definido.

Así, la máquina computa la función g definida de modo que

$$g(n_1,\ldots,n_m) = (j_1,\ldots,j_k)$$

siempre que la máquina arranca con:

- $[i] = n_{i+1},\ i = 0,\ldots,m-1,$
- $[i] = 0,\ \forall i \geq m$

y termina con

- $[i] = j_{i+1},\ i = 0,\ldots,k-1$

El modelo RASP

La RASP (Random Access Stored Program Machine) es como una RAM pero con el programa a ejecutar almacenado en la propia memoria; así su arquitectura se corresponde con la de von Neumann.

Es una máquina universal que puede simular a cualquier RAM y en consecuencia realizar todas sus computaciones. Es en este sentido similar a la máquina de Turing Universal (véase el tema 5).

La Máquina Contador

El modelo de la `Máquina Contador` es una restricción de la RAM. Sólo dispone de un número finito de registros (que naturalmente pueden variar de una máquina a otra) y no tiene instrucciones con direccionamiento indirecto.

El resto de esta sección lo dedicaremos a su estudio exponiendo diversos ejemplos operativos.

Este modelo de máquina aunque presente estas restricciones tiene el poder computacional de la RAM y de la RASP.

Su conjunto básico de instrucciones es el siguiente:

- `suc(i)`, su semántica es la siguiente: `incrementa en una unidad el contenido de` R_i, es decir,

  $[i] \leftarrow [i]+1$.

  La ejecución continúa en la siguiente instrucción del programa si existe, si no termina.

- `pre(i,k)`, su semántica es la siguiente:

  - si $[i]>0$, entonces $[i] \leftarrow [i]-1$; la ejecución continúa en la siguiente instrucción del programa si existe, si no termina;
  - en otro caso, la ejecución continúa en la k-ésima instrucción si existe, si no termina.

Este par de instrucciones junto con la existencia de un registro predefinido con contenido nulo son suficientes para realizar cualquier computación.

Seguidamente veremos algunos fragmentos de programa que ejecutan cálculos sencillos que ilustran la operatividad de este modelo.

En lo inmediato que sigue R_C tiene asignado permanentemente el valor nulo.

- **[i] ← 0**

```
  n: pre(i,n+2)
n+1: pre(c,n)
```

- **[i] ← [i]+[j] ∧ [j] ← 0 ∧ $R_i \neq R_j$**

```
  n: pre(j,n+3)
n+1: suc(i)
n+2: pre(c,n)
```

Para evitar el tener que mantener un registro predefinido permanentemente con valor nulo añadiremos al conjunto de instrucciones la instrucción

- `goto(n)`, su semántica es la siguiente: `ejecutar incondicionalmente la instrucción n-ésima si ésta existe, si no terminar la ejecución.`

Las instrucciones suc(i), pre(i,k) y goto(n) son suficientes para realizar cualquier computación.

Los ejemplos anteriores, utilizando la sentencia `goto(n)`, quedan como sigue.

- **[i] ← 0**

```
  n: pre(i,n+2)
n+1: goto(n)
```

Este código puede simbólicamente referenciarse mediante la macroinstrucción `cer(i)`.

- **[i] ← [i]+[j] ∧ [j] ← 0 ∧ $R_i \neq R_j$**

```
  n: pre(j,n+3)
n+1: suc(i)
n+2: goto(n)
```

- **[i] ← k**

```
     cer(i)
n+1: suc(i)
.....
.....
n+k: suc(i)
```

Este código puede simbólicamente referenciarse mediante la macroinstrucción `asi(k,i)`.

- **[i] ← [j] ∧ $R_i \neq R_j$**

  Sea R_k un registro auxiliar de cálculo distinto de los anteriores: $R_i \neq R_k \wedge R_j \neq R_k$.

```
      cer(i)
      cer(k)
  n:  pre(j,n+4)
n+1:  suc(i)
n+2:  suc(k)
n+3:  goto(n)
n+4:  pre(k,n+7)
n+5:  suc(j)
n+6:  goto(n+4)
```

Este código puede simbólicamente referenciarse mediante la macroinstrucción `cop(j,i)`.

- **[m] ← [p]+[q]**

  Sean R_i y R_j dos registros auxiliares de cálculo distintos de los anteriores.

```
      cop(p,i)
      cop(q,j)
  n:  pre(j,n+3)
n+1:  suc(i)
n+2:  goto(n)
n+3:  cop(i,m)
```

Este código puede simbólicamente referenciarse mediante la macroinstrucción `sum(p,q,m)`.

- **[r] ← [p].[q]**

  Sean R_i, R_j, R_k y R_m cuatro registros auxiliares de cálculo distintos de los anteriores. (Se puede prescindir de R_j).

```
      cer(m)
      cer(k)
      cop(p,i)
      cop(q,j)
  n:  pre(i,n+8)
n+1:  pre(j,n+5)
n+2:  suc(m)
n+3:  suc(k)
n+4:  goto(n+1)
n+5:  pre(k,n)
n+6:  suc(j)
n+7:  goto(n+5)
n+8:  cop(m,r)
```

Este código puede simbólicamente referenciarse mediante la macroinstrucción `mul(p,q,m)`.

Equivalentemente, utilizando la macroinstrucción para la suma

```
        cer(m)
        cop(p,i)
   n:   pre(i,n'+1)
        sum(m,q,m)
   n':  goto(n)
n'+1:   cop(m,r)
```

- **[r] ← [p]/[q]**

  Sean R_i, R_j, R_k y R_m cuatro registros auxiliares de cálculo distintos de los anteriores.

```
      cer(m)
      cer(k)
      cop(p,i)
      cop(q,j)
  n:  pre(j,error)
n+1:  suc(j)
n+2:  pre(j,n+6)
n+3:  pre(i,n+10)
n+4:  suc(k)
n+5:  goto(n+2)
n+6:  suc(m)
n+7:  pre(k,n+2)
n+8:  suc(j)
n+9:  goto(n+7)
n+10: cop(m,r)
  ..............

error:......
```

Este código puede simbólicamente referenciarse mediante la macroinstrucción `div(p,q,m)`.

- **if([p] ≤ [q]) then go to m1**
  else go to m2

Sean R_i y R_j dos registros auxiliares de cálculo distinto de los anteriores.

```
     cop(p,i)
     cop(q,j)
  n: pre(i,m1)
n+1: pre(j,m2)
n+2: goto(n)
```

Este código puede simbólicamente referenciarse mediante la macroinstrucción `mei(p,q,m1,m2)`.

- **if([i] = [j]) then go to m1**
  else go to m2

Sean R_i y R_j dos registros auxiliares de cálculo distinto de los anteriores.

```
     cop(p,i)
     cop(q,j)
  n: pre(i,n+3)
n+1: pre(j,m2)
n+2: goto(n)
n+3: pre(j,m1)
n+4: goto(m2)
```

Este código puede simbólicamente referenciarse mediante la macroinstrucción `igu(p,q,m1,m2)`.

- **while([i]>0)CÓDIGO**

```
  n: pre(i,m+1)
n+1: suc(i)
     CÓDIGO
  m: goto(n)
```

Equivalencias entre modelos

Sea una función (parcial) de la forma

$$g: N^m \longrightarrow N^k$$

Diremos que la función g es:

- RAM computable si y sólo si existe una RAM que la compute.
- RASP computable si y sólo si existe un programa para la máquina RASP con el que esta máquina puede computarla.
- MC computable si y sólo si existe una máquina contador que la compute.

Teorema: Sea una función (parcial) de la forma

$$g: N^m \longrightarrow N^k$$

Los siguiente enunciados son equivalentes:

- g es RAM computable.
- g es RASP computable.
- g es MC computable.
- g es Turing computable.

Sea una función (parcial) de la forma

$$g: N \longrightarrow N$$

Diremos que la función `g` es: `M2C computable` si y sólo si existe una máquina contador con dos contadores que la compute.

Teorema: Sea una función (parcial) de la forma

$$g: N \longrightarrow N$$

Los siguiente enunciados son equivalentes:

- `g` es `MC` computable.
- `g` es `M2C` computable.

Observación: El poder computacional del modelo de máquina contador con dos contadores es también equivalente al de los modelos:

- RAM,
- RASP, y
- Turing.