Lenguajes de Programación
Nota de clase 2: el lenguaje PostFix*

Favio E. Miranda Perea Lourdes Del Carmen González Huesca
Facultad de Ciencias UNAM

15 de agosto de 2018

Introducimos aquí el lenguaje PostFix que nos servirá como herramienta de estudio en diversas partes del curso. Se trata de un lenguaje simple basado en una pila e inspirado en el lenguaje gráfico PostScript, el lenguaje de programación Forth (`http://www.complang.tuwien.ac.at/projects/forth.html`)y las calculadoras HP.

1. Sintaxis

- La unidad sintáctica básica de un programa PostFix es el comando. Un comando es de alguna de las siguientes formas:
  - Una literal entera: `3 -8, 0,...`
  - Una de las siguientes palabras reservadas:

    `add,div,eq,exec,gt,lt,mul,nget,pop,rem,sel,sub,swap`
  - Una secuencia ejecutable: una secuencia parentizada de comandos separados por espacios en blanco:

    `(7 add), (8 add 3 swap), (2 (5 mul) exec add),...`
- Programas: un programa en PostFix es una secuencia parentizada compuesta de:
  1. La palabra reservada `postfix`.
  2. Un número natural `n` que indica el número de argumentos de entrada.
  3. Una sucesión posiblemente vacía de comandos.
- Ejemplos de programas:

```
(postfix 0 4 7 sub)
(postfix 5 add 7 add div)
(postfix 2 nget 3 pop (exec 5 swap))
```

Es importante recalcar que todos los paréntesis en PostFix son indispensables y omitirlos o cambiarlos de posición influye en el comportamiento del programa. Por ejemplo las siguientes tres secuencias ejecutables son todas distintas:

*Esta nota se basa en el capítulo 1 del libro *Design Concepts in Programming Languages. F. Turbak, D. Gifford y M.A. Sheldon. MIT Press 2008.*

```
((1) (2 3 4) swap exec)

((1 2) (3 4) swap exec)

((1 2 (3 4 swap) exec)
```

2. Semántica

- El significado de un programa se determina ejecutando sus comandos en orden de izquierda a derecha.
- Cada comando manipula una pila de valores implícita que inicialmente contiene a los argumentos enteros del programa, el primero en el tope y el último en el fondo de la pila.
- Un valor en la pila es una literal entera o bien una secuencia ejecutable.
- El resultado de un programa es el valor entero que queda en el tope de la pila una vez que su secuencia de comandos ha sido ejecutada por completo.
- Un programa causa error si sucede alguna de las siguientes situaciones:
  - Al inicio de la ejecución el número de elementos en la pila y el número de argumentos de entrada son distintos.
  - Al final de la ejecución la pila está vacía.
  - Al final de la ejecución el valor en el tope de la pila no es un entero.
  - La pila es inapropiada para la ejecución de un comando.

2.1. Semántica de los comandos

- `n`: agrega el elemento `n` a la pila
- `sub`: Si el primer (tope) y segundo elementos de la pila son v_1, v_2 respectivamente, eliminarlos de la pila y agregar $v_2 - v_1$ en su lugar.
- `add,mul,div,rem`: son los operadores de suma, producto, división entera y residuo (módulo) y se definen análogamente a `sub`.
- `lt`: Si el primer (tope) y segundo elementos de la pila son v_1, v_2 respectivamente, eliminarlos de la pila. Si $v_2 < v_1$ agregar un 1 a la pila, en otro caso agregar un 0 a la pila.
- `eq,gt`: son los operadores de igualdad y mayor que y se definen análogamente a `lt`.
- `pop`: elimina el elemento que está en el tope de la pila.
- `swap`: intercambia los primeros dos elementos en el tope de la pila.
- `sel`: Si el primer (tope), segundo y tercer elementos de la pila son v_1, v_2, v_3 respectivamente, eliminarlos de la pila. Si v_3 es el número 0 entonces agregar v_1 a la pila; si v_3 es un número distinto de 0, agregar v_2 a la pila.
- `nget`: llamemos v_{ind} al tope de la pila y $v_1, \ldots, v_n$ a los elementos restantes en orden siendo v_n el que está en el fondo. Eliminar v_{ind} de la pila. Si v_{ind} es un número j tal que $1 \leq j \leq n$ y v_j es un número, agregar v_j a la pila.

- Secuencia ejecutable (`c1 c2...cn`): agregar la secuencia ejecutable como un solo valor a la pila. Las secuencias ejecutables se usan en conjunción con `exec`
- `exec`: eliminar la secuencia ejecutable del tope de la pila y agregar en orden sus comandos al inicio de la secuencia de comandos actualmente en ejecución.

2.2. Ejemplos de ejecución

- Programa: `(postfix 0 -1 2 add 3 mul)`
- Ejecución:
  - Pila inicial: [ ]

    ```
    [ ]
    [-1]
    [2,-1]
    [1]
    [3,1]
    [3]
    ```

  - Pila final: [3]
- Resultado: 3
- Programa: `(postfix 3 mul swap 2 mul swap sub)`
- Ejecución:
  - Pila inicial: [5,4,3]

    ```
    [5,4,3]
    [20,3]
    [3,20]
    [2,3,20]
    [6,20]
    [20,6]
    [-14]
    ```

  - Pila final: [-14]
- Resultado: -14
- Programa: `(postfix 1 (2 mul) exec)`
- Ejecución:
  - Pila inicial: [7]

    ```
    [7]
    [(2 mul), 7]
    [7]
    [2,7]
    [14]
    ```

- Pila final: [14]

- Resultado: 14

- Programa: `(postfix 0 (0 swap sub) 7 swap exec)`

- Ejecución:

  - Pila inicial: [ ]

    ```
    [ ]
    [(0 swap sub)]
    [7,(0 swap sub)]
    [(0 swap sub), 7]
    [7]
    [0,7]
    [7,0]
    [-7]
    ```

  - Pila final: [-7]

- Resultado: -7

- Programa: `(postfix 4 lt (add) (mul) sel exec)`

- Ejecución:

  - Pila inicial: [5,6,4,3]

    ```
    [5,6,4,3]
    [0,4,3]
    [(add),0,4,3]
    [(mul),(add),0,4,3]
    [(mul), 4,3]
    [4,3]
    [12]
    ```

  - Pila final: [12]

- Resultado: 12

- Programa: `(postfix 2 2 nget)`

- Ejecución:

  - Pila inicial: [9,12]

    ```
    [9,12]
    [2,9,12]
    [12,9,12]
    ```

  - Pila final: [12,9,12]

- Resultado: 12

2.3. Errores de ejecución

Se observa que hay muchos casos donde la ejecución puede causar un error y detenerse abruptamente. Para que una implementación sea robusta estos errores deben ser manejados. La manera más simple es terminar la ejecución con un mensaje informativo en el caso de que la pila sea inapropiada para la ejecución del comando actual. Los casos generales son:

- La pila tiene menos elementos que los necesarios para una operación. Por ejemplo, si se quiere ejecutar `add` y la pila sólo tiene un elemento o si la pila tiene menos de tres números al tratar de ejecutar `sel`
- Los argumentos para la ejecución del comando actual no son adecuados. Por ejemplo, si se quiere ejecutar `sel` y el tercer elemento en la pila no es un número, o si se quiere ejecutar `nget` y el tope de la pila no es un número.

Algunos ejemplos de programas erroneos son:

- `(postfix 2 swap)`, pila inicial: `[3]`
- `(postfix 1 pop)`, pila inicial: `[4,5]`
- `(postfix 1 4 mul add)`, pila inicial: `[3]`
- `(postfix 2 4 sub div)`, pila inicial: `[4,5]`
- `(postfix 2 3 nget)`, pila inicial: `[7,8]`
- `(postfix 1 (2 mul) 1 nget)`, pila inicial: `[3]`
- `(postfix 0 (2 mul) )`, pila inicial: `[ ]`
- `(postfix 0 (2 mul) 3 4 sel)`, pila inicial: `[ ]`

Dejamos como ejercicio indicar el error particular en cada caso.

3. Desventajas de la semántica informal

La definición anterior del lenguaje PostFix es muy similar a la de los manuales de lenguajes de programación reales. Sin embargo esta clase de descripciones son inadecuadas si se quiere estudiar formalmente un lenguaje. Mencionemos algunos problemas potenciales:

- Mejoramiento de programas transformando código complejo en frases más simples y eficientes. ¿Cómo asegurarnos que el proceso de transformación preserva el significado de un programa ?
- Prueba de propiedades del lenguaje: por ejemplo como probar que todo programa en PostFix termina, es decir no causa ciclos infinitos.
- Extensiones: cómo asegurar que cierta extensión del lenguaje, por ejemplo con variables, arreglos o ciclos, preserva o viola cierta propiedad original como la terminación o la preservación de tipos. Con cada característica nueva la descripción informal crece y se vuelve más complicado razonar acerca de la interacción de diversas características, algunas de las cuales resultan ortogonales mientras que otras modifican las propiedades del lenguaje al combinarse.

- Implementaciones: dada la especificación informal cómo asegurar que distintos intérpretes del lenguaje se comportan exactamente igual al ejecutar cualquier programa. La especificación informal resultará inevitablemente ambigua y distintos equipos de implementación podrían resolver la ambigüedad de manera distinta.
- Sobreespecificación: la descripción informal en nuestro caso se basa en la secuencia de comandos y en la pila de valores. Los programadores de un intérprete podrían pensar que el uso de la pila es obligatorio. Aunque en una implementación directa la pila es necesaria, no tiene porque ser indispensable. De hecho existe una implementación de PostFix que no usa pila.

En las siguientes clases nos dedicaremos al estudio de semánticas formales para distintos prototipos de lenguajes de programación similares a PostFix.