Lenguajes de Programación 2019-I[*]
Nota de clase 3: Definiciones Inductivas[**]

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

15 de agosto de 2018

Existen dos formas de definir un lenguaje de programación, una consiste en describir en lenguaje natural, en este caso español, la clase de expresiones legales del lenguaje así como la forma en que éstas se evalúan. Esta técnica tiene la ventaja de que el lector puede fácilmente familiarizarse con el lenguaje pero los detalles acerca del mismo son difíciles de extraer a partir de estas descripciones.

Otra manera es usar un metalenguaje formal para describir y razonar acerca de diversas características y propiedades del lenguaje, por ejemplo:

- Gramática del lenguaje.
- Reglas de alcance.
- Sistemas de tipos (semántica estática)
- Estrategias de evaluación (semántica dinámica (operacional))

Esta técnica tiene como ventajas la especificación clara y detallada del lenguaje lo cual lo convierte en un lenguaje analizable formalmente.

El metalenguaje usado no necesita ser un lenguaje eficiente o susceptible de ejecutarse puesto que su propósito primario es explicar el lenguaje objeto a los usuarios o a los diseñadores de intérpretes o compiladores del mismo. De hecho podriamos usar como metalenguaje directamente a la lógica y la teoría de conjuntos sobre un universo de texto de programa. Es decir, los cómputos en abstracto pueden describirse en términos de relaciones sobre texto.

El metalenguaje que usaremos es el de las definiciones inductivas mediante reglas de inferencia, similares a las reglas lógicas de deducción natural, que describimos a continuación.

1. Objetos y Juicios

Las nociones fundamentales en la clase de definiciones inductivas que trataremos son objetos y juicios.

- Objetos: se consideran como primitivos y previamente dados. Por ejemplo números, árboles, tipos, valores, etc.

[*]Agradecemos la colaboración de Susana Hahn Martin Lunas en la revisión 2018-I

[**]Estas notas se basan en diversas versiones del libro de Robert Harper *Practical Foundations of Programming Languages*

- Juicios: Un juicio es una afirmación acerca de un objeto particular cumple cierta propiedad o bien que dos o más objetos pertenecen a una relación. Por ejemplo:

n Nat	n es un número natural
$n = m + k$	n es la suma de m con k
t asa	t es un árbol sintáctico
$3 + 4 * 5$ ExpAr	$3 + 4 * 5$ es una expresión aritmética válida
0.1234 float	0.1234 es un valor flotante
"aba" pal	"aba" es una cadena palíndrómo
T type	T es un tipo
$e : T$	La expresión e tiene tipo T
$e \to e'$	la expresión e se reduce a e'
$e \Downarrow v$	la expresión e se evalua al valor v.

Por lo general usaremos notación infija de la forma $s\ P$ para expresar que s cumple el juicio P o notación infija, cuando el juicio involucra a dos objetos, como $n = k, e : T, e \to e'$. Obsérvese que los juicios son esencialmente predicados en lógica, escritos de manera postfija, por ejemplo en lugar de $P(s)$ escribimos $s\ P$.

2. Reglas de Inferencia

Una regla de inferencia es un esquema de la forma

$$\frac{J_1 \quad J_2 \ldots J_n}{J}$$

donde J_i, J son juicios.

Ya hemos manejado esta clase de reglas en lógica. Los juicios $J_1 \ldots J_n$ son las *premisas* y J es la *conclusión* de la regla. Si no hay premisas la regla es un *axioma*. Algunos ejemplos son:

$$\frac{}{0\ \text{Nat}} \qquad \frac{n\ \text{Nat}}{suc(n)\ \text{Nat}}$$

$$\frac{}{1\ \text{impar}} \qquad \frac{n\ \text{impar}}{suc(suc(n))\ \text{impar}}$$

$$\frac{n\ \text{par} \quad m\ \text{par}}{n + m\ \text{par}} \qquad \frac{n\ \text{par} \quad m\ \text{impar}}{n + m\ \text{impar}}$$

$$\frac{p\ \text{varp}}{p\ \text{form}} \qquad \frac{\varphi\ \text{form} \quad \psi\ \text{form}}{\varphi \vee \psi\ \text{form}}$$

$$\frac{\varphi\ \text{true} \quad \psi\ \text{true}}{\varphi \wedge \psi\ \text{true}} \qquad \frac{\varphi\ \text{true}}{\varphi \vee \psi\ \text{true}}$$

3. Definiciones Inductivas

Una regla de inferencia $\dfrac{J_1 \;\; J_2 \ldots J_n}{J}$ es inductiva si al menos uno de los juicios J_i es de la misma forma que el juicio J, es decir, ambos se refieren a la misma propiedad o relación.

Una definición inductiva es un conjunto finito de reglas de inferencia donde al menos una de ellas es inductiva. Por ejemplo considérese la siguiente definición de listas:

- nil es una lista de elementos de A
- Si a es un elemento de A y ℓ es una lista de elementos de A entonces $cons(a, \ell)$ es una lista de elementos de A.
- Son todas.

Esta definición inductiva corresponde a la siguiente definición mediante reglas de inferencia:

$$\frac{}{nil \;\; \mathsf{list}_A} \qquad \frac{x \; A \quad \ell \;\; \mathsf{list}_A}{cons(a, \ell) \quad \mathsf{list}_A}$$

Mas adelante puntualizaremos a que corresponde, en la definición por reglas de inferencia, la cláusula *son todas* en la definición original.

Como otro ejemplo veamos las definiciones inductivas de las propiedades de ser un número natural y un número natural par, definidas aquí con los juicios n Nat, n Even:

$$\frac{}{0 \;\; \mathsf{Nat}} \qquad \frac{n \;\; \mathsf{Nat}}{\mathsf{s}(n) \;\; \mathsf{Nat}}$$

$$\frac{}{0 \;\; \mathsf{Even}} \qquad \frac{n \;\; \mathsf{Even}}{\mathsf{s}(\mathsf{s}(n)) \;\; \mathsf{Even}}$$

- Las definiciones inductivas nos sirven para definir relaciones de manera mecánica.
- Es importante notar que no todas las relaciones pueden definirse mediante una definición inductiva. Por ejemplo la relación de ser una lista infinita.
- Sin embargo, la mayoría de relaciones necesarias para el estudio de lenguajes de programación se sirven de una definición inductiva.

4. Derivaciones

- Para mostrar que una instancia de una relación definida inductivamente es válida basta mostrar una derivación de dicha instancia.
- Una derivación es una composición o encadenamiento de reglas de inferencia a partir de axiomas la cual termina con la instancia que se quiere demostrar.
- Una derivación tiene una estructura de árbol donde la derivación de las premisas de una regla son los hijos de un nodo que representa una instancia de dicha regla.

- Dichos árboles se suelen desarrollar con la raíz hasta abajo.

Veamos un par de ejemplos:

- $\mathsf{s}(\mathsf{s}(\mathsf{s}\,0)))\ \mathsf{Nat}$

$$\dfrac{\dfrac{\dfrac{0\ \mathsf{Nat}}{\mathsf{s}(0)\ \mathsf{Nat}}}{\mathsf{s}(\mathsf{s}(0))\ \mathsf{Nat}}}{\mathsf{s}(\mathsf{s}(\mathsf{s}(0)))\ \mathsf{Nat}}$$

- $cons(1, cons(2, nil))\ \mathsf{list}_{\mathsf{Nat}}$

$$\dfrac{1\ \mathsf{Nat} \quad \dfrac{2\ \mathsf{Nat} \quad \dfrac{nil\ \mathsf{list}_{\mathsf{Nat}}}{cons(2,nil)\ \mathsf{list}_{\mathsf{Nat}}}}{cons(1,(cons(2,nil))\ \mathsf{list}_{\mathsf{Nat}}}$$

Existen dos formas principales para construir derivaciones de un juicio dado:

- *Encadenamiento hacia adelante o construcción de abajo hacia arriba*: se inicia con los axiomas siendo la meta el juicio deseado. En este caso se mantiene un conjunto de juicios derivables que inicialmente es vacío, extendiendolo con la conclusión de cualquier regla cuyas premisas ya están en el conjunto. El proceso termina cuando el juicio deseado entra al conjunto. Este es un método indirecto en el sentido de que no se toma en cuenta el juicio meta al decidir como proceder en cada paso. Si el juicio es derivable entonces la aplicación exhaustiva del proceso terminará eventualmente con la conclusión deseada, en caso contrario es imposible en general decidir cuando parar en la construcción del conjunto y concluir que el juicio no es derivable.
- *Encadenamiento hacia atrás o construcción de arriba hacia abajo*: Se inicia con el juicio deseado buscando encadenar reglas hasta terminar en axiomas. Esta búsqueda mantiene una serie de metas actuales que son juicios cuyas derivaciones se buscan. Inicialmente este conjunto contiene únicamente el juicio deseado. En cada etapa se elimina un juicio del conjunto de metas y consideramos todas las reglas cuya conclusión es dicho juicio. Para cada regla agregamos sus premisas al conjunto de metas. El proceso termina cuando el conjunto de metas es vacío. Si el juicio es derivable eventualmente terminará este proceso. Sin embargo en el caso contrario, no hay en general un algoritmo para decidir que el juicio no es derivable.

5. Formalización de la sintaxis del lenguaje PostFix

- Comandos:
  - *Literales enteras:*

$$\dfrac{m \in \mathbb{Z}}{m\ LitE}\ (le)$$

  - *Palabras reservadas:*

$$\dfrac{}{\texttt{add}\ PalR}(pradd) \ldots \dfrac{}{\texttt{swap}\ PalR}(prswap)$$

  - *Comandos:*

$$\dfrac{c\ LitE}{c\ Com}(comlit) \ldots \dfrac{c\ PalR}{c\ Com}(compal)$$

- *Secuencia de comandos:* El juicio $s\ SeqC$ formaliza la definición de una secuencia posiblemente vacía de comandos $s = c_1 \dots c_n$ o bien $s = \varepsilon$ (secuencia vacía)

$$\frac{}{\varepsilon\ SeqC}\ (sc1) \qquad \frac{c\ Com \quad s\ SeqC}{c\ s\ SeqC}\ (sc2)$$

- *Secuencia ejecutable:* El juicio $s\ SeqE$ formaliza la definición de una secuencia ejecutable no vacía de comandos $s = (c_1 \dots c_n)$

$$\frac{s\ SeqC \quad s \neq \varepsilon}{(s)\ SeqE}\ (se)$$

Veamos algunas derivaciones:

- `(2 add -3 swap)` $SeqE$

1.	`2` Com		
2.	`add` Com		
3.	$-$`3` Com		
4.	`swap` Com		
5.	$\varepsilon\ SeqC$	$(sc1)$	
6.	`swap` $\varepsilon\ SeqC$	$(sc2)$	$4, 5$
7.	$-$`3 swap` $\varepsilon\ SeqC$	$(sc2)$	$3, 6$
8.	`add` $-$ `3 swap` $\varepsilon\ SeqC$	$(sc2)$	$2, 7$
9.	`2 add` $-$ `3 swap` $\varepsilon\ SeqC$	$(sc2)$	$1, 8$
10.	(`2 add` $-$ `3 swap` ε) $SeqE$	(se)	9

- `(7 mul) exec (sel) pop` $SeqC$

1.	`7` Com		
2.	`mul` Com		
3.	$\varepsilon\ SeqC$	$(sc1)$	
4.	`mul` $\varepsilon\ SeqC$	$(sc2)$	$2, 3$
5.	`7 mul` $\varepsilon\ SeqC$	$(sc2)$	$1, 4$
6.	(`7 mul` ε) $SeqE$	(se)	5
7.	`exec` Com		
8.	`sel` Com		
9.	`sel` $\varepsilon\ SeqC$	$(sc2)$	$8, 3$
10.	(`sel` ε) $SeqE$	(se)	
11.	`pop` Com		
12.	`pop` $\varepsilon\ SeqC$	$(sc2)$	$11, 3$
13.	(`sel`) `pop` $\varepsilon\ SeqC$	$(sc2)$	$10, 12$
14.	`exec` (`sel`) `pop` $\varepsilon\ SeqC$	$(sc2)$	$7, 13$
15.	(`7 mul`) `exec` (`sel`) `pop` $\varepsilon\ SeqC$	$(sc2)$	$6, 14$

6. Inducción

La gran mayoría de las demostraciones en este curso se harán usando inducción estructural sobre alguna definición inductiva, dicho principio puede resumirse como sigue:
Suponga que una propiedad o relación A está definida inductivamente por un conjunto de reglas de inferencia S_A. Para mostrar que una segunda propiedad P es válida para todos los elementos x de A basta probar que para cualquier regla que pertenezca a S_A, digamos

$$\frac{x_1\ A \ldots x_n\ A}{x\ A}$$

se cumple que:

$$\textit{Si } P \textit{ es válida para } x_1, \ldots, x_n \textit{ entonces } P \textit{ es válida para } x.$$

Es decir, basta mostrar que cada regla

$$\frac{x_1\ P \ldots x_n\ P}{x\ P}$$

obtenida al sustituir A por P en las reglas de S_A, es una regla válida.

Observemos que las reglas de S_A que son axiomas corresponden a los casos base de la inducción mientras que las otras reglas corresponden a pasos inductivos donde las premisas corresponden a la hipótesis de inducción.

Algunos ejemplos específicos son:

- Números naturales Nat: el principio usual de inducción para naturales es el siguiente:
  - Base de la inducción: probar $P(0)$
  - Hipótesis de inducción (H.I.): suponer $P(x)$
  - Paso inductivo: probar, usando la H.I., que $P(\mathsf{s}(x))$

  lo cual en forma de reglas corresponde a :

$$\frac{}{0\ P} \qquad \frac{n\ P}{\mathsf{s}(n)\ P}$$

- Listas list_A:
  - Base de la inducción: probar $P(nil)$
  - H.I. suponer $a\ in A$ y $P(\ell)$
  - Paso inductivo: probar, usando la H.I., $P((a : \ell))$

  lo cual en forma de reglas corresponde a :

$$\frac{}{nil\ P} \qquad \frac{a\ A \quad \ell\ P}{(a : \ell)\ P}$$

- Números pares Even:
  - Base de la inducción: probar $P(0)$.
  - H.I. suponer $P(n)$
  - Paso inductivo: probar $P(\mathsf{s}(\mathsf{s}(n)))$

  lo cual en forma de reglas corresponde a :

$$\frac{}{0\ P} \qquad \frac{n\ P}{\mathsf{s}(\mathsf{s}(n))\ P}$$

7. Lenguajes de paréntesis balanceados

En esta sección desarrollamos una formalización del lenguaje de paréntesis balanceados para ejemplificar a detalle los conceptos discutidos hasta ahora e introducir algunos conceptos relacionados de importancia.

El lenguaje de paréntesis balanceados a considerar es

$$M = \{\varepsilon, (), (()), \ldots, ()(), ()(()), (()()(())), \ldots\}$$

Este lenguaje es un ejemplo clásico de lenguaje libre de contexto definido mediante la siguiente gramática en forma BNF

$$M ::= \varepsilon \mid (M) \mid MM$$

Esta definición puede reescribirse mediante reglas de inferencia como sigue:

$$\frac{}{\varepsilon\ M}(m1) \qquad \frac{s\ M}{(s)\ M}(m2) \qquad \frac{s_1\ M \qquad s_2\ M}{s_1 s_2\ M}(m3)$$

aquí $(m1), (m2)$ y $(m3)$ son nombres para las reglas, útiles para referencias posteriores.

El significado del juicio $s\ M$ es que s es una cadena de paréntesis balanceados. Es decir, s es una cadena balanceada de paréntesis si y sólo si existe una derivación de $s\ M$ mediante las reglas $(m1), (m2), (m3)$.

Es importante observar que el número de reglas en una definición inductiva debe ser mínimo, esto facilitará las pruebas inductivas al haber menos casos que verificar en el paso inductivo. Respecto a este punto una vez que se tiene un conjunto de reglas primitivas o básicas Φ debemos distinguir otras clases de reglas, las derivables y las admisibles.

- *Reglas derivables*: Una regla $\mathcal{R}$ es derivable respecto a un conjunto de reglas básicas $\Phi = \{\mathcal{R}_1, \ldots, R_k\}$ si y sólo si $\mathcal{R}$ puede obtenerse usando las reglas primitivas de Φ, es decir si hay una derivación de K que utilice únicamente reglas de Φ al suponer $J_1, \ldots, J_n$. Formalmente, la regla

$$\frac{J_1 \ldots J_n}{K}$$

  es derivable si al suponer válidos los juicios $J_1, \ldots, J_k$ podemos concluir el juicio K mediante reglas de Φ. Por ejemplo la regla

$$\frac{s\ M}{((s))\ M}$$

es derivable con respecto a $\Phi_M = \{(m1), (m2), (m3)\}$ porque se tiene la siguiente derivación parcial que utiliza la regla $(m2)$ dos veces.

$$\dfrac{\dfrac{s\ M}{(s)\ M}}{((s))\ M}$$

- *Reglas admisibles* Una regla es admisible respecto a Φ si y sólo si el hecho de que sus premisas sean derivables a partir de Φ implica que su conclusión también es derivable a partir de Φ. Es decir, la regla

  $$\dfrac{J_1 \ldots J_n}{K}$$

  es admisible si cada vez que podemos derivar $J_1 \ldots J_n$ entonces necesariamente derivaremos K.
  Puede observarse que una regla admisible no cambia el contenido del lenguaje, es decir, no genera mas cadenas. Por ejemplo la regla

  $$\dfrac{()s\ M}{s\ M}$$

  es admisible con respecto a Φ_M pues si derivamos $()s\ M$ entonces usamos la regla $(m3)$ necesariamente pero esto implica que tuvimos que derivar primero $s\ M$ que es lo que necesitabamos para probar la admisibilidad de la regla.

Obsérvese que una regla derivable es admisible mas no al revés. La regla anterior no es derivable pues no tenemos en las reglas primitivas una regla que elimine expresiones.
En contraste la siguiente regla

$$\dfrac{(s)\ M}{s\ M}$$

no es admisible con respecto a Φ_M pues podemos derivar por ejemplo $()()\ M$ pero no $)(\ M$.

Para afianzar los conceptos veamos unos ejemplos con números pares e impares. Las reglas primitivas son:

$$\dfrac{}{0\ \mathsf{par}}\ (0p) \qquad \dfrac{n\ \mathsf{par}}{suc(n)\ \mathsf{impar}}\ (si) \qquad \dfrac{n\ \mathsf{impar}}{suc(n)\ \mathsf{par}}\ (sp)$$

Por cierto, esta es una definición inductiva simultánea, estamos definiendo dos juicios par e impar al mismo tiempo dependiendo uno del otro.
Se deja como ejercicio verificar que la siguiente regla es derivable

$$\dfrac{n\ \mathsf{par}}{suc(suc(n))\ \mathsf{par}}\ (ssp)$$

Además la siguiente regla es admisible pero no derivable:

$$\dfrac{suc(n)\ \mathsf{impar}}{n\ \mathsf{par}}\ (invsi)$$

Veamos por qué: no es derivable pues no hay una regla que pase de impar a par eliminando una aplicación de sucesor. Es admisible pues si derivamos $suc(n)$ impar necesariamente fue mediante la

regla (si) de manera que también derivamos n par.

Debemos enfatizar que las nociones de derivabilidad y admisibilidad dependen siempre de un conjunto fijo de reglas primitivas Φ. Al cambiar éste los conceptos pueden cambiar de igual forma.

Sigamos ahora con el lenguaje de paréntesis balanceados y analicemos algunas propiedades importantes.

Proposición 1 *Si se cumple $s\ M$, entonces s tiene el mismo número de paréntesis izquierdos que derechos.*

Demostración. Inducción sobre las reglas.

- Base: $(m1)$. Claro pues entonces $s = \varepsilon$ y hay cero paréntesis izquierdos y derechos.
- Paso inductivo para $(m2)$. $s = (s')$. Por HI s' tiene la misma cantidad n de paréntesis izquierdos que de paréntesis derechos, lo cual claramente implica que s mantiene esta propiedad con $n+1$ paréntesis izquierdos y la misma cantidad de paréntesis derechos.
- Paso inductivo para $(m3)$. $s = s_1 s_2$. Por HI s_1 tiene el mismo número de paréntesis izquierdos y derechos, digamos n_1. Análogamente por HI, s_2 tiene n_2 paréntesis izquierdos y derechos. Entonces s tiene $n_1 + n_2$ paréntesis izquierdos y derechos.

⊣

Obsérvese que la gramática original para M es ambigua, una característica indeseable para una gramática de un lenguaje de programación pues se pueden obtener árboles sintácticos no únicos, es decir que una cadena en el lenguaje sea resultado de dos o más árboles. Recordemos que eliminar la ambigüedad de una gramática no es un problema algorítmico de manera que las soluciones son artesanales. En este caso podemos proponer otra gramática representada con las siguientes reglas:

$$\frac{}{\varepsilon\ L}\ (l1) \qquad \frac{s_1\ L \quad s_2\ L}{(s_1)s_2\ L}\ (l2)$$

El objetivo ahora es mostrar que ambas definiciones son equivalentes, es decir, $s\ M$ si y sólo si $s\ L$.

Empezamos con un lema necesario:

Lema 1 (Concatenación en L) *Si $s_1\ L$ y $s_2\ L$ entonces $s_1 s_2\ L$. Es decir, la regla*

$$\frac{s_1\ L \quad s_2\ L}{s_1 s_2\ L}$$

es admisible.

Demostración. Inducción sobre $s_1\ L$. El lector debe verificar que una inducción sobre $s_2\ L$ no funcionaría en este caso.

- Base de la inducción $(l1)$. En este caso $s_1 = \varepsilon$. Queremos probar entonces que $\varepsilon s_2\ L$, es decir, $s_2\ L$ pero esto es una premisa.

- Paso inductivo ($l2$). Tenemos $s_1 = (s_{11})s_{12}$ con $s_{11}\ L$, $s_{12}\ L$ y queremos probar $s_1 s_2\ L$. Por HI obtenemos $s_{12}s_2\ L$ de manera que la regla ($l2$) nos permite concluir $(s_{11})s_{12}s_2\ L$ que es nuestra meta.

⊣

Para probar la equivalencia probamos cada parte por separado.

Lema 2 *Si $s\ M$ entonces $s\ L$*

Demostración. Inducción sobre $s\ M$. ⊣

Lema 3 *Si $s\ L$ entonces $s\ M$*
Demostración. *Inducción sobre $s\ L$.* ⊣

7.1. Analizador sintáctico mediante definiciones inductivas

Hasta ahora el uso de definiciones inductivas ha sido para especificar tipos de datos o lenguajes mediante gramáticas. En el caso del lenguaje de paréntesis la especificación es el juicio $s\ M$. Sin embargo, tambien es necesario implementar un algoritmo de reconocimiento para el lenguaje, es decir, un analizador sintáctico. Esto puede hacerse, por ejemplo, mediante un autómata de pila. La idea es clara, se procesa la cadena guardando en la pila los paréntesis izquierdos y borrando del tope de la pila un paréntesis izquierdo si se está leyendo un paréntesis derecho en la cadena. La cadena se aceptará al consumir todos los símbolos quedando la pila vacía.
Este proceso puede expresarse mediante una definición inductiva del juicio $\langle k, s\rangle\ \mathsf{pila}$ cuyo significado intuitivo es: se acepta s al haber k paréntesis izquierdos en la pila. Es decir, s es válida al haber k paréntesis izquierdos guardados en la pila.
La definición inductiva es la siguiente:

- Se acepta ε si hay 0 paréntesis en la pila. De haber paréntesis en la pila, ya no habría forma de aparearlos con el resto de la cadena a saber ε y por tanto la cadena no puede aceptarse.

$$\frac{}{\langle 0, \varepsilon\rangle\ \mathsf{pila}}\ (p1)$$

- Se acepta la cadena (s con k paréntesis en la pila si se acepta la cadena s con $k+1$ paréntesis en la pila. El símbolo a leer es (el cual se guardará en la pila, quedarán $k+1$ paréntesis en la pila y se debe procesar s.

$$\frac{\langle k+1, s\rangle\ \mathsf{pila}}{\langle k, (s\ \rangle\ \mathsf{pila}}\ (p2)$$

- Se acepta la cadena $)s$ con k paréntesis en la pila si $k > 0$ y se acepta s con $k-1$ paréntesis en la pila. Puesto que el símbolo a leer es) observemos que si $k = 0$ el paréntesis derecho no se podría aparear. Al leer), este paréntesis se apareó con un paréntesis izquierdo de la pila por lo que quedan $k-1$ y se debe procesar s.

$$\frac{\langle k-1, s\rangle\ \mathsf{pila} \quad k > 0}{\langle k,\ )s\ \rangle\ \mathsf{pila}}\ (p3)$$

Ahora debemos probar formalmente que esta definición inductiva funciona como analizador sintáctico, es decir, sólamente se aceptan las cadenas balanceadas. Para ello requerimos de algunos pasos intermedios que son los siguientes lemas:

Lema 4 *Si* $\langle k_1, s_1\rangle$ pila, $\langle k_2, s_2\rangle$ pila *entonces* $\langle k_1 + k_2, s_1 s_2\rangle$ pila.
Demostración. *Inducción sobre* $\langle k_1, s_1\rangle$ pila ⊣

Lema 5 *Si* $s\ M$ *entonces* $\langle 0, s\rangle$ pila
Demostración. *Inducción sobre* $s\ M$ ⊣

Lema 6 *Si* $l\,r\ M$ *y* $c\ M$ *entonces* $l\,c\,r\ M$
Demostración. *Inducción sobre* $l\,r\ M$*. Esta inducción es algo intrincada. Obsérvese que la regla* $(m2)$ *no es invertible (por ejemplo se tiene que* $()()\ M$ *y claramente ni* $(\ M$ *ni* $)()\ M$*.). Lo mismo sucede con la regla* $(m3)$ *por lo que los pasos inductivos requieren de un análisis de caso.*

- *Base de la inducción, regla* $(m1)$*: en este caso* $l\ r = \varepsilon$ *por lo que* $l = r = \varepsilon$ *y así* $lcr = c$*. Por lo tanto por hipótesis se tiene* $lcr\ M$*.*
- *Paso inductivo, regla* $(m2)$*: existen tres casos:*
  1. $l = \varepsilon$ *y* $r = (r_1)$*. Análogo al siguiente caso.*
  2. $l = (l_1)$ *y* $r = \varepsilon$*. En este caso se tiene que* $lr = (l_1)$ *y entonces por hipótesis* $(l_1)\ M$*. Así* $lcr = (l_1)c$ *y aplicando* $(m3)$ *a* $(l_1)\ M$ *y* $c\ M$ *se sigue que* $lcr\ M$*.*
  3. $l = (l_1$ *y* $r = r_1)$*. En este caso se tiene que* $lr = (l_1 r_1)$ *y como* $lr\ M$ *entonces se sigue que* $l_1 r_1\ M$ *puesto que* $lr\ M$ *se obtuvo mediante* $(m2)$*. De manera que por la H.I. se sigue que* $l_1 c r_1\ M$ *y por* $(m2)$ *concluimos que* $(l_1 c r_1)\ M$*, es decir* $lcr\ M$*.*
- *Paso inductivo, regla* $(m3)$*: existen dos casos:*
  1. $l = l_1 l_2$ *con* $l_1\ M$ *y* $l_2 r\ M$*. Por H .I. se sigue que* $l_2 cr\ M$ *y finalmente como* $l_1\ M$ *por* $(m3)$ *se sigue que* $l_1 l_2 cr\ M$*, es decir,* $lcr\ M$*.*
  2. $r = r_1 r_2$ *con* $r_1\ M$ *y* $r_2\ M$*. Es análogo.*

  ¿ por qué no es necesario el caso en que $l\ M$ *y* $r\ M$ *?*

⊣

Lema 7 *Si* $\langle k, s\rangle$ pila *entonces* $\underbrace{(\ldots(}_{k}\ s\ M$*.*

Demostración. *Inducción sobre* $\langle k, s\rangle$ pila ⊣

Ahora si, formalmente debemos probar que $s\ M$ *si y sólo si* $\langle 0, s\rangle$ pila*:*

Proposición 2 $s\ M$ *si y sólo si* $\langle 0, s\rangle$ pila.
Demostración. *Directo de los lemas 5 y 7 con* $k = 0$*.* ⊣