Teoría de Lenguajes

Primer cuatrimestre 2013

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

TP Micro HTML Prettyprint

Integrante	LU	Correo electrónico
Ezequiel Gutesman	715/02	egutesman@gmail.com
Mauricio Alfonso	65/09	mauricioalfonso88@gmail.com
Victor Hugo Montero	707/98	vico.walker@gmail.com

1 Introducción

Cuando comenzamos a pensar en cómo encarar el problema del desarrollo de la gramática, pensamos que podríamos tratar cada caracter (letra, número, espacio, etc.) como un símbolo terminal diferente, pero terminaríamos con una cantidad inmanejable de símbolos no terminales y producciones, es decir con una gramática enorme. Por esa razón definimos una serie de Token Léxicos que al pasárselos a un analizador léxico transforman el texto de entrada de una serie de caracteres en una serie de tokens.

Definimos los tokens usando expresiones regulares. Primero definimos un token por cada tag de HTLM y por último un token para todo el texto que sobre. El analizador léxico tokeniza el archivo buscando los tokens en orden, de manera que todo el texto que halla entre dos tags queda tokenizado como `texto`.

El caso de la sección `script` de HTML nos presentó un problema: entre los tags `<script>` y `</script>` puede haber cualquier combinación de caracteres excepto el tag de cierre de script. Esto quiere decir que podría haber incluso otros tags HTML dentro de un script, que deberían ser ignorados y que podrían no formar HTML válido. Pensamos en un principio en tokenizar el texto interior de los scripts con el analizador léxico como `texto_sin_script` para tomar todo el texto que no tenga el tag `</script>` como un token. Sin embargo no es posible tokenizar el texto de esta manera, ya que no tiene sentido poner el token `texto_sin_script` antes ni después de los tags de HTML. Esto se debe a que si el analizador léxico busca primero el token `texto_sin_script` y luego los tags, el primer token estaría "absorviendo" todos los tags excepto `</script>` como parte de un script, incluso cuando no forman parte de un script; entonces no tokenizaría nunca los tags HTML. Si en cambio tokenizáramos primero los tags HTML y después `texto_sin_script`, los tags del interior de los scripts serían reconocidos y el tag `texto_sin_script` no cumpliría su función. Por esta razón decidimos tokenizar únicamente `texto` para todo el texto que sobre luego de haber reconocido los tags HTML, y reconocer los tags que pueda haber en el script desde la gramática y no desde el analizador léxico.

Además de tokenizar, el analizador léxico también se encarga de eliminar los comentarios y los espacios en blanco consecutivos, ya que estos podrían estar en cualquier parte del archivo recibido y debemos ignorarlos por completo.

En la gramática definimos un símbolo no terminal S para todo el documento.

El símbolo no terminal H contiene el interior del HTML, que a su vez puede tener un $HEAD$ y un $BODY$.

Los símbolos $HEAD$ y $BODY$ representan las secciones head y body de HTML respectivamente, y los símbolos HE y B representan el interior de dichas secciones.

Con el símbolo SCS definimos una serie de cero o más scripts, que identificamos con el símbolo SC.

El símbolo TSC representa el texto del interior de un script. Este es el símbolo con más producciones, ya que puede tener en su interior cualquier tag excepto `</script>` y además estos pueden ir de cualquier manera, sin formar HTML válido.

El interior de la sección body lo definimos recursivamente con el símbolo B, entendiendo que entre cada par de tags de apertura y cierre del mismo tipo puede haber más texto HTML válido.

Decidimos que sólo puede haber un elemento `<TITLE>...</TITLE>` dentro del head.

2 Tokens Léxicos

Los tokens léxicos están descritos por expresiones regulares. La siguiente tabla describe para cada token su expresión rgular correspondiente.

Token Léxico	*Expresión Regular*
`<HTML>`	`<html>`
`</HTML>`	`</html>`
`<HEAD>`	`<head>`
`</HEAD>`	`</head>`
`<BODY>`	`<body>`
`</BODY>`	`</body>`
`<TITLE>`	`<title>`
`</TITLE>`	`</title>`
`<SCRIPT>`	`<script>`
`</SCRIPT>`	`</script>`
`<DIV>`	`<div>`
`</DIV>`	`</div>`
`<P>`	`<p>`
`</P>`	`</p>`
` `	` `
`texto`	`.*`

Antes de tokenizar, el analizador léxico elimina espacios consecutivos y comentarios. Los comentarios cumplen con la expresión regular `<!--.*-->`

3 Gramática

La siguiente gramática define el MicroHTML:

$$G =< N, \Sigma, P, S >$$

Donde:

$$N = \{S, H, HEAD, BODY, HE, SCS, SC, TSC, B\}$$
$$\Sigma = \{texto, < HTML >, < /HTML >, < HEAD >, < /HEAD >,$$
$$< TITLE >, < /TITLE >, < SCRIPT >, < /SCRIPT >,$$
$$< BODY >, < /BODY >, < H1 >, < /H1 >, < DIV >, < /DIV >,$$
$$< P >, < /P >, < BR >\}$$
$$S = S$$

Que de acuerdo a la clasificación de Chomsky es una gramática *tipo 0*, o *sin restricciones*. Las producciones (P) se detallan en la siguiente subsección.

3.1 Producciones - P

S → *<HTML> H </HTML>*

H → *HEAD BODY* | *HEAD* | *BODY* | λ

HEAD → *<HEAD> HE </HEAD>*

BODY → *<BODY> B </BODY>*

HE → *SCS <TITLE> texto </TITLE> SCS*

SCS → *SC SCS* | λ

SC → *<SCRIPT> TSC </SCRIPT>*

TSC →
<HTML> TSC | *</HTML> TSC* |
<HEAD> TSC | *</HEAD> TSC* |
<BODY> TSC | *</BODY> TSC* |
<TITLE> TSC | *</TITLE> TSC* |
<DIV> TSC | *</DIV> TSC* |
<H1> TSC | *</H1> TSC* |
<P> TSC | *</P> TSC* |
<SCRIPT> TSC | *
 TSC* | *texto TSC* | λ

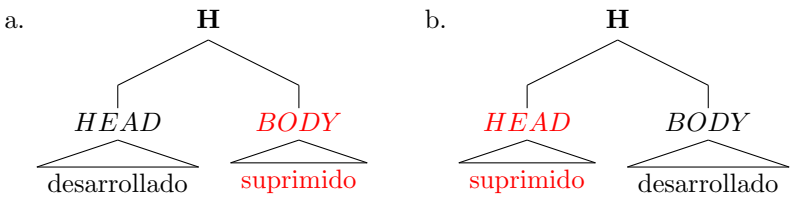
B →
texto B |
<DIV> B </DIV> B |
<H1> B </H1> B |
<P> B </P> B |
*
 B* |
λ

4 Arbol de derivación

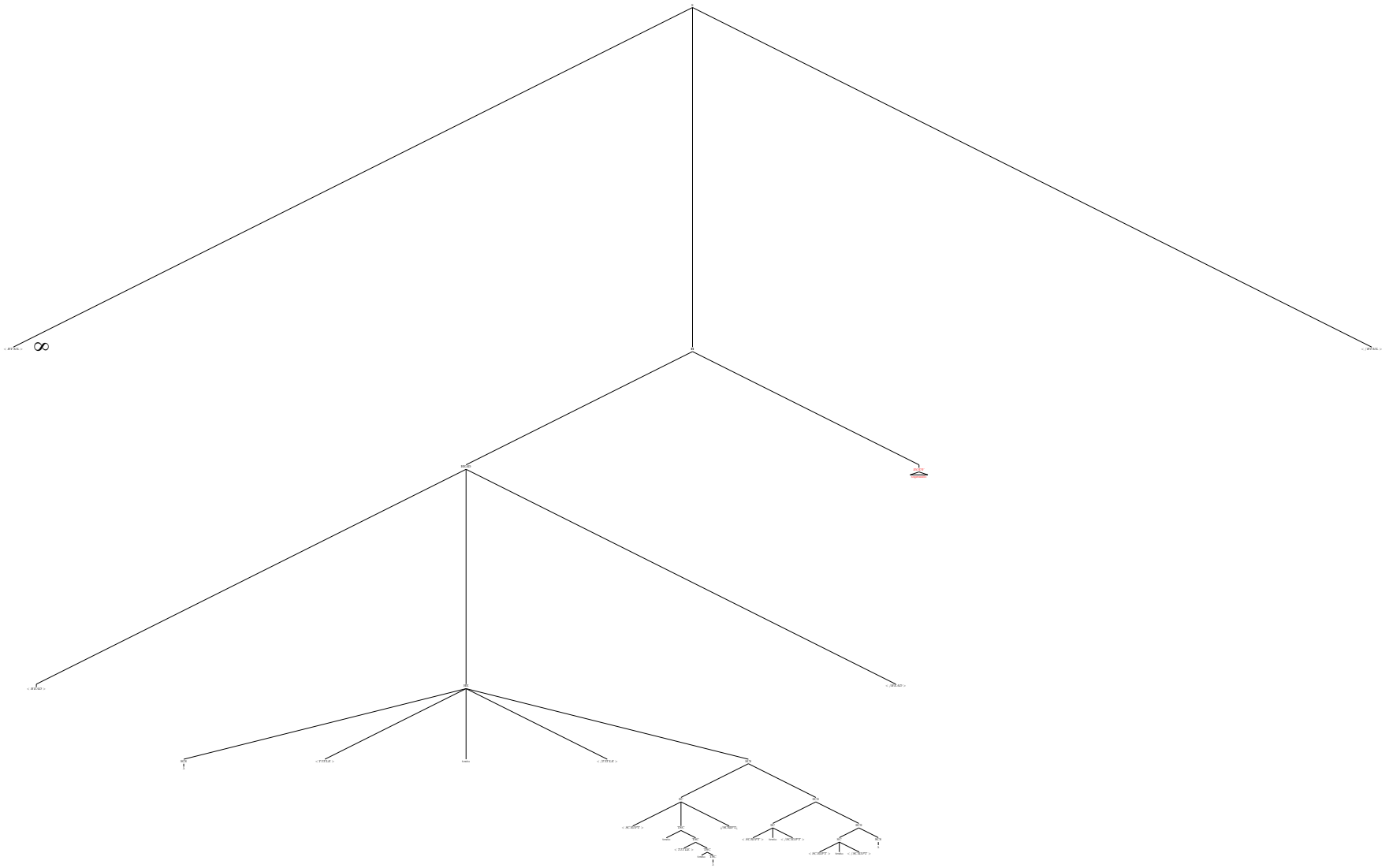
A continuación presentamos el árbol de derivación para el siguiente ejemplo:

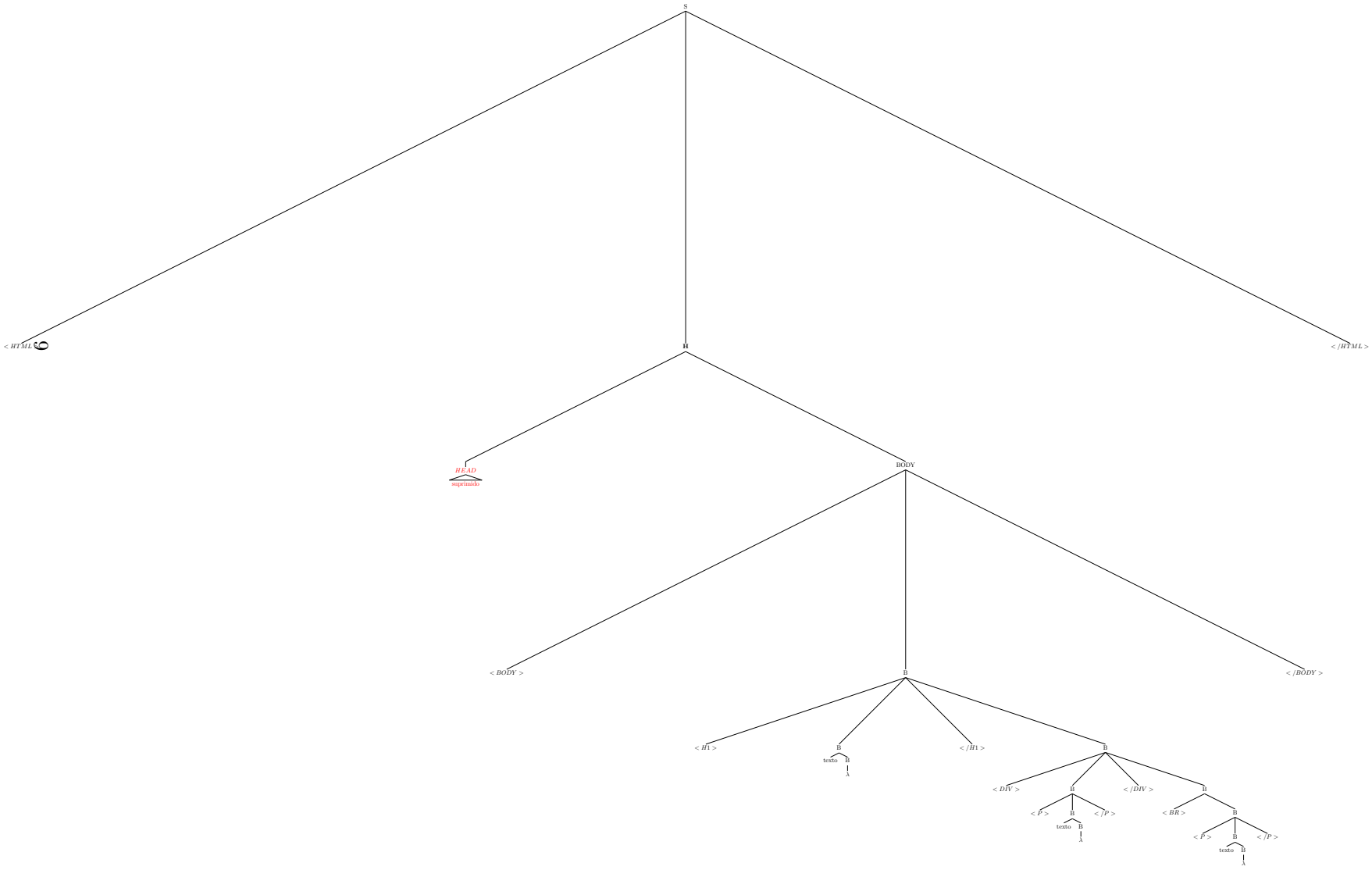
```
<HTML>
    <HEAD>
        <TITLE>Una pagina de ejemplo</TITLE>
        <SCRIPT>
        function unaFunc(){
            alert("esta funcion imprime un tag roto <TITLE>");
        }
        </SCRIPT>
        <SCRIPT></SCRIPT>
        <SCRIPT>alert("aca no aparece el cierre de SCRIPT")</SCRIPT>
    </HEAD>
    <BODY>
        <H1>Un heading</H1>
        <DIV>
            <P>Este texto es de prueba</P>
        </DIV>
        <BR>
        <P>Mas prueba</P>
    </BODY>
</HTML>
```

Debido al tamaño del ejemplo, el árbol de derivación tuvo que ser partido en dos partes. Nótese que del (único) nodo con el no terminal H cuelgan 2 hijos con el no-terminal $HEAD$, y el no terminal $BODY$. El primer subárbol muestra los descendientes de $HEAD$ y el segundo los de $BODY$. En cada árbol, el no terminal cuyo subárbol está suprimido se encuentra marcado en rojo, y con un subárbol genérico (triangulo) simbolizando que en realidad continúa. Por ejemplo:



En la figura *a.* se suprime el subárbol corrspondiente al no terminal $BODY$ y en el *b.* el correspondiente al no terminal $HEAD$.




S
< HTML >
H
< /HTML >
HEAD
suprimido
BODY
< BODY >
B
< /BODY >
< H1 >
B
texto
B
λ
< /H1 >
B
< DIV >
B
< /DIV >
< P >
B
texto
B
λ
< /P >
B
< BR >
B
< P >
B
texto
B
λ
< /P >